DECISION
SCIENCE
CONSORTIUM, INC.

AD-A266 360

(2)

TESTING AND EVALUATING C^3I SYSTEMS THAT EMPLOY AI

(CLIN 0001)

VOLUME 3: A GUIDE TO DEVELOPING SMALL EXPERT SYSTEMS

Decision Science Consortium, Inc.
1895 Preston White Drive, Suite 300
Reston, Virginia 22091

January 1991

Final Report
Period of Performance: 16 September 1988 - 15 September 1990
Contract Number: DAEA18-88-C-0028
PR&C Number: W61DD3-8057-0601
AAP Number: EPG 8048

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

Prepared for:
U.S. Army Electronic Proving Ground
ATTN: STEEP-ET-S (Mr. Robert J. Harder)
Fort Huachuca, Arizona 85613-7110

TECHNICAL REPORT 90-9

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) 90-9	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Decision Science Consortium, Inc. 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION US Army Electronic Proving Ground STEEP-ET-S
6c. ADDRESS (City, State, and ZIP Code) 1895 Preston White Drive, Suite 300 Reston, Virginia 22091	7b. ADDRESS (City, State, and ZIP Code) Ft. Huachuca, Arizona 85613-7110
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable) STEEP-ET-S	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DAEA-18-88-C-0028
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification) TESTING AND EVALUATING C^3I SYSTEMS THAT EMPLOY AI -- VOLUME 3: A GUIDE TO DEVELOPING SMALL EXPERT SYSTEMS

12. PERSONAL AUTHOR(S) ICF Information Technology, Inc.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final Technical	FROM Sep 88 TO Sep 90	1991 January 31	116

16. SUPPLEMENTARY NOTATION The views, opinions, and/or findings contained in this report are those of the authors and should not be construed as an official Department of the Army position, policy, or decision unless so designated by other documentation.

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Expert Systems, Testing, Knowledge-Based Systems, Artificial Intelligence, Multiattribute Utility

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The "Guide to Developing Small Expert Systems" provides an introduction to expert systems, then guides the user through the phases of expert system development: Concept, Definition Prototype Development, Test and Evaluation, Final System Development, and Post-Development. Each phase is fully defined, and milestones and other important issues are discussed. The user is guided through the development process by a series of checklists and worksheets which are to be completed during each phase. Sample forms have been filled out and included throughout the document to serve as an example. Blank forms are also provided which can be reproduced for use in actual system development.

A hypertext version of the document is also available on floppy disk. This version was implemented using interactive software, and includes the full text, graphics, pop-up windows, reference links, and other useful features.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Mr. Robert J. Harder	22b. TELEPHONE (Include Area Code) (602) 538-2090 — 22c. OFFICE SYMBOL STEEP-ET-S

DD Form 1473, JUN 86 Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

GUIDE TO DEVELOPING SMALL EXPERT SYSTEMS

PREPARED BY
ICF INFORMATION TECHNOLOGY, INC.

SUPPORTED BY
U.S. ARMY ELECTRONIC PROVING GROUND
UNDER CONTRACT NUMBER DAEA 18-88-C-0028

February 1991

CONTENTS

FIGURES

Foreword

The *Guide to Developing Small Expert Systems* was developed for the U.S. Army Electronic Proving Ground, in support of their expert system development program. It is Volume 3 of ***Testing and Evaluating C^3I Systems That Employ AI***, a report that was also prepared for the U.S. Army. Another important part of this report is the **MAU Framework for Testing and Evaluating Expert Systems**. The framework and a brief explanation of its components are included as Appendix A. Although the **MAU Framework** was designed for larger expert systems, several key concepts can also be used quite successfully in the development of small expert systems, and will be referenced throughout this document. Together, these documents provide an excellent means of reference for those interested in developing expert systems.

Objectives

Expert system development is a learning process. Ideally, it is also a continuous process, as the success of one expert system motivates the development of future systems. This powerful technology is increasingly finding a place in organizations, as people realize the tremendous potential of expert systems, and as users and experts begin to take the initiative in expert system development. By following the guidelines set forth in this guidebook, an organization is well on its way to developing successful small expert system applications. This document provides the basic guidelines for expert system development, and is also intended to increase the level of interest in expert systems throughout the U.S. Army.

Primary Users

This document is a guidance tool that is designed to show the user **HOW** to develop an expert system application. It is intended to be used by U.S. Army personnel who are:

- New to expert system development
- Targeting a small PC-based system
- Developing with an expert system tool or "shell."

The users of this document will be developing small expert systems, consisting of approximately one hundred rules. The typical development team will be small, ranging from one to three people. The intended user group may range from a small group of users within the organization to an agency-wide group of users located around the country.

How to Use the Guide

The main body of the text consists of a step-by-step guide through the phases of expert system development. It will touch on the main concepts of expert systems, but will focus on the practical development process. As each step in the development process is described, it is also illustrated through the use of checklists and worksheets accompanying each phase.

The checklist for each phase summarizes the major tasks to be completed or decisions to be made in order for the milestone for that phase to be achieved. Some of the phases also use a supplementary worksheet, that contains detailed breakdowns of the issues in the checklist. Throughout the text, you will find samples of these forms, filled out as they would be during actual expert system development. A blank set of forms is also provided for copying purposes.

The presence of supporting worksheets is indicated on the checklists by a numbering system (e.g., C.1, C.2, etc.). When these numbers precede an item on a checklist, additional steps are required and the related section on the worksheet must be completed before the item can be checked on the checklist.

The checklists and worksheets incorporate several means for completing a task, including the following:

- Two columns of boxes labeled "YES" and "NO" (see Fig. 2.2). The boxes in each column are given point values of 0 or 1. Put a check mark in the appropriate boxes and add up the total number of points for each item. Use the numeric value for each box to determine that the task has been completed satisfactorily.

- A LOW -> HIGH ranking system (see Fig. 2.1). Put a check mark on the line that is most appropriate. In some cases, each column is assigned a point value of 1 to 5. Add up the total number of points based on the numbering system shown.

- A single box to check off when the task has been completed, and a line to indicate the date of completion (see Fig. 4.1). For this type of evaluation, a check mark indicates the satisfactory completion of the task.

- A numeric ranking system (see Fig. 4.2). Each of the items should be numbered in order of importance, with 1 being the most important.

In order to guide you through the development of an expert system, a sample expert system application, called **ESTEL**, is included in this guidebook. Each of the checklists, worksheets, and supporting documents have been completed for **ESTEL** to illustrate the steps taken during the development of an actual expert system. A brief explanation of each of these is also provided. **ESTEL** is a simplified version of a more complex expert system application, **XTEL**, that was developed for the Defense Communications Agency to be used in the design of military telecommunications networks (Liebowitz, 1988).

1.0 An Introduction to Expert Systems

An expert system is a software application that contains knowledge about a specific subject area, or domain, and uses reasoning mechanisms to perform tasks in a manner similar to a human expert. Such systems are also referred to as knowledge-based systems, advisory systems, and expert problem-solvers. These alternative terms are useful in describing systems that involve a specific body of knowledge or a decision process, but are not intended to replicate an expert's performance.

Expert systems are an application tool belonging to the field of computer science known as Artificial Intelligence (AI). Expert systems are not a new development; they have been in existence since the late 1960's. But only recently have they come into the mainstream of computing systems. This is due primarily to recent hardware and software advances. The advent of the Personal Computer (PC) and its accompanying software has made expert system development accessible to a larger number of people. Other improvements in hardware capabilities, such as increased memory capacity, enhanced graphics, and ease of portability, have also contributed to the increased popularity of expert systems.

An important advance is the availability of expert system "shells," such as CLIPS and EXSYS. These are off-the-shelf expert system development tools, which provide a ready-made interface and built-in inferencing mechanisms. An expert system shell can be compared to a spreadsheet package, which has a number of built-in functions, and allows a user without an accounting background to produce a useful spreadsheet. Many expert system shells are menu-driven, or use an English-based format for rule development, requiring no previous programming experience. The abundance of expert system shells has made this technology available to people who do not have experience in a programming language.

1.1 Types of Expert Systems

Expert systems represent a very versatile technology that can be used to address a broad range of application areas and uses. Generally, expert systems are designed to serve in one of the following capacities:

- **Advisory** - acts as an assistant to skilled users; role is to remind user of known information and facts, suggest promising lines of research, gather and analyze data from external sources and generally support the user in performing a task.

- **Process Control** - monitors an on-line or real-time process (e.g., production line) and provides feedback to adjust, refine, or correct processing.

- **Training** - acts as an information source to novices in a skill area, providing detailed explanation facilities and guidance with a focus

on knowledge transfer rather than problem solving; a tutorial system may also diagnose the user's errors and provide lessons to correct them

- **Problem Solving** - focuses on gathering pertinent information and providing a specific solution to a problem, often supplying extensive justification; may also try to minimize resource use (time, number of questions asked).

- **Trouble-shooting** - also known as fault isolation - a form of diagnosis aimed at determining which component of a system is causing an error, once an error has been detected.

- **Diagnosis** - classification type of problem that uses a set of symptoms to lead to a conclusion or diagnosis; often used to identify problems.

- **Design** - focuses on providing a solution to a set of specifications and constraints, usually emphasizing a minimization of some feature such as cost, size, or assembly time.

- **Configuration** - similar to design, takes a given set of components and suggests a "good" means of combining them; may also identify missing components.

- **Repair** - given a system error, proposes a solution to the problem; often used following a diagnostic system, it may emphasize minimizing cost or speed to recovery.

Expert system applications can be developed in nearly any subject area, and systems are currently being used to do everything from performing equipment maintenance and diagnosis to simplifying government regulations. The chosen subject area is not as important as selecting an appropriate problem and limiting the scope of the system, as you will learn in the **Concept Phase**.

1.2 Expert System Development Phases

Developing a successful expert system application is not a simple task. It requires the support and cooperation of users, management, experts, and developers. This guidebook will lead you through the phases of the development process, focusing on the important issues to consider, the common problems that are encountered, and critical milestones along the way. The importance of testing the system throughout the development process is also emphasized. The major phases of expert system development are:

- **Concept Phase** - The problem area is selected and analyzed to determine if an expert system is the appropriate tool to use. The feasibility of an expert system application is studied, and team support of the project is initiated.

- **Definition Phase** - The scope and purpose of the expert system is carefully defined. The **Requirements Statement**, incorporating the general purpose, capabilities, and expectations of the expert system, is written. A development team is selected, users are consulted, and an initial design is produced.

- **Prototype Development Phase** - Developers gather expertise in the subject area and convert the knowledge into logic in the selected development tool. A prototype expert system is developed with input from the users and expert. Documentation of the system is initiated.

- **Test and Evaluation Phase** - The expert system is verified and validated by experts and developers. Usefulness, interface, and documentation issues are addressed by users.

- **Final Development Phase** - Comments from testing are analyzed and incorporated into the expert system. Revised version is retested and prepared for distribution. Documentation is revised and finalized.

- **Post-Development Phase** - The system is distributed to users, and training and technical support are provided. Long-term maintenance issues are addressed.

The development process for a small expert system will range from six to twelve weeks in length. Variable factors include the availability of team members to work on the project, and the team's level of familiarity with expert systems and the subject area. Fig. 1.1 indicates the approximate percentage of the total development time that should be spent on each of the phases. These should not be regarded as exact figures, but rather as a guideline.

Throughout the phases, it is important to keep in mind that expert system development is an iterative, repetitive process, as shown in Fig. 1.2. During each phase, decisions made earlier should be re-evaluated, and the data and system operation should be tested. For example, as knowledge is being acquired and system development begins during the **Prototype Development Phase**, issues often surface which make it necessary to revise the design or functions of the expert system. When this occurs, the development team should revisit the decisions made during the **Definition Phase**, and make appropriate changes to the **Requirements Statement** and system design. Phases can also be repeated as the problem becomes more clearly defined.

Frequent revisions and re-evaluations during the development process should not be interpreted as mismanagement of the project or a failure to adhere to the original requirements. On the contrary, this sort of activity is encouraged and indicates that the team is interested in developing an effective expert system. Because expert systems are applied to problems involving qualitative reasoning or judgment, most successful systems are significantly refined two to three times during the development process.

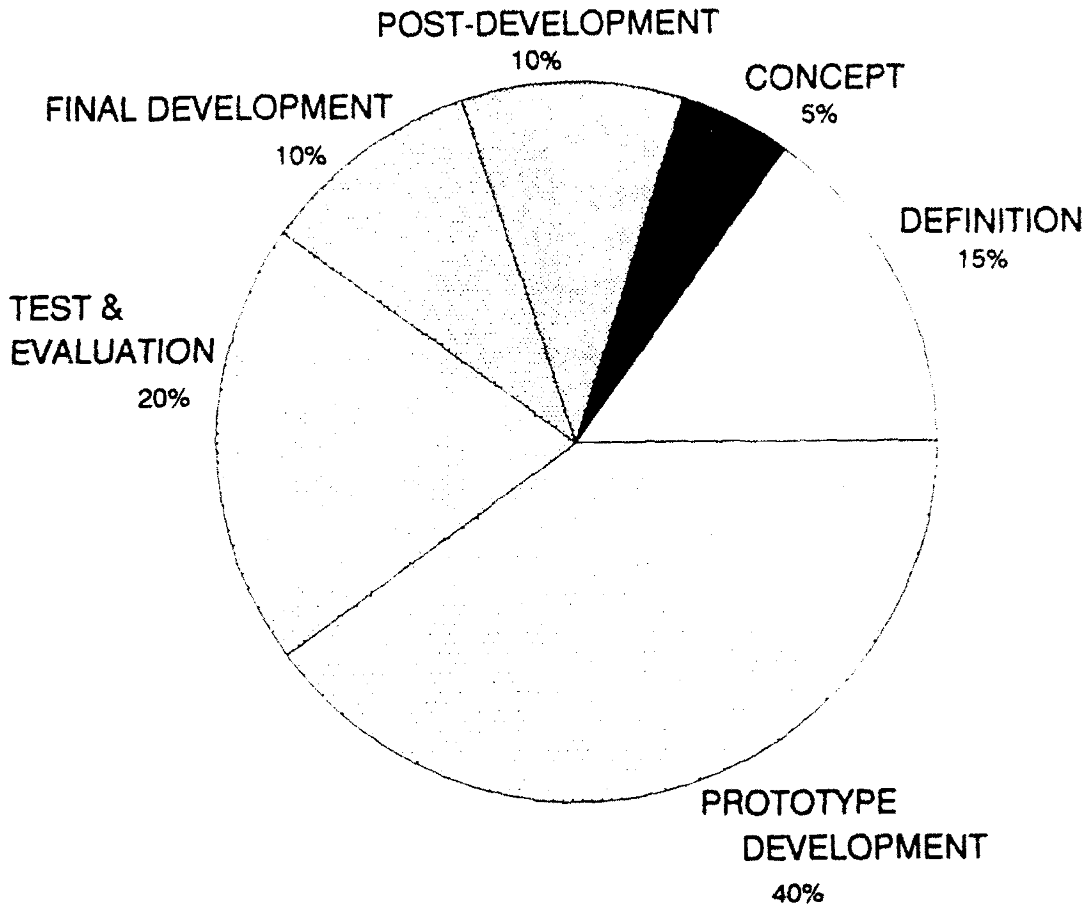

Fig. 1.1 Percentage of Time Spent on Each Phase

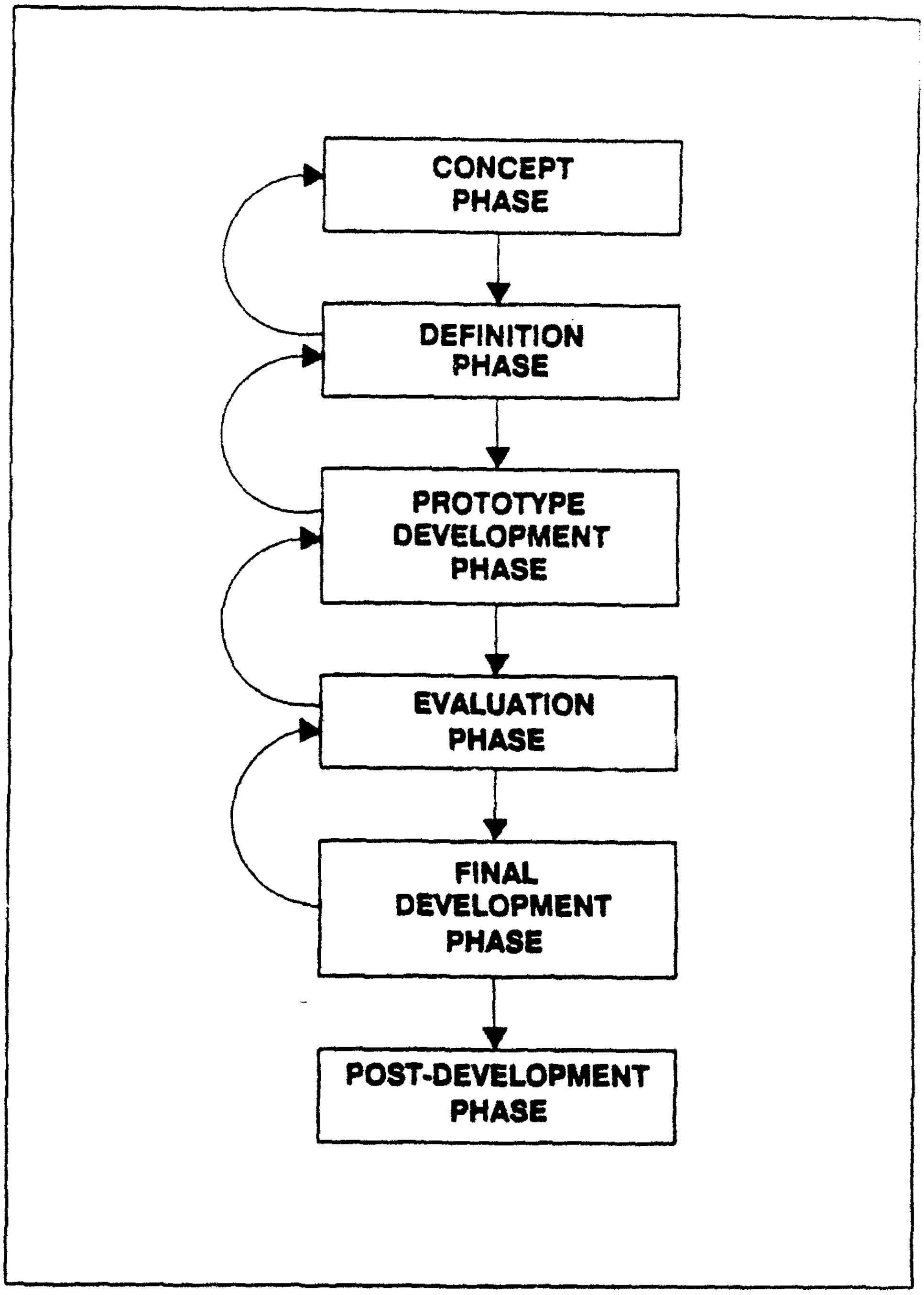

Fig. 1.2 Expert System Development Phases

1.3 Development Milestones and Critical Factors

The milestones involved in the expert system development process build on one another, as shown in Fig. 1.3, with each step providing a solid base for the next. The critical milestones during the development process include:

- Selecting expert system technology for the application
- Producing the **Requirements Statement**
- Producing the expert system design
- Selecting the development team
- Producing the prototype expert system
- Testing and evaluating the expert system
- Producing the final version of the expert system
- Implementing the expert system.

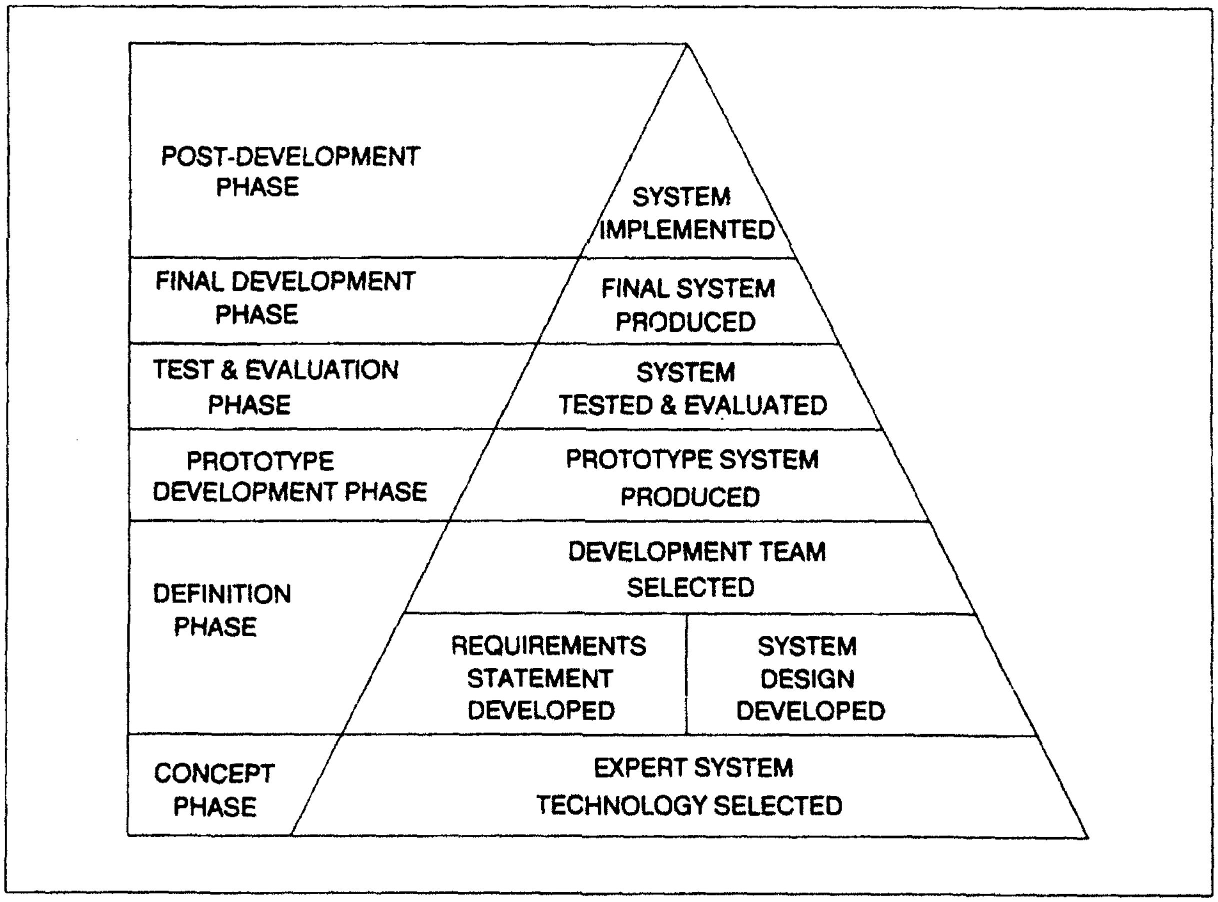

Fig. 1.3 Expert System Development Milestones

Although common development problems will be addressed in each section, it is important to mention three critical factors at this time that will have a major impact on the development of a successful expert system, and will be mentioned throughout the development phases:

- *Testing*
- Internal support
- User involvement.

The first important issue is the testing of the expert system. Testing is not a one-time event, but occurs throughout the development process. A number of steps can also be taken during the development process that will enhance the ease of full system testing during the **Test and Evaluation Phase**. These items will be mentioned throughout the document. The expert, developers, and users should all be actively involved in testing, in order to fully evaluate the rules, inferencing processes, and interface, respectively. Throughout this document, references will be made to the **MAU Framework**, which discusses methods for testing expert systems.

Second, support of the expert system's development by management or a proponent of the project is vital to the production of a successful application. Management should be aware from the outset of the intricacies and issues of expert system development, and also the need for cooperation and patience in the effort.

The third important issue is to involve the users throughout the development process. Users are an important link in every phase of development, because they can be instrumental in guiding the project. Users can provide valuable insights into alternative means to fulfill the system design, system objectives, and user interface issues. It is important to keep in mind that a successful expert system application is one that is placed in production and actually used. Working with the intended users is the best way to ensure this.

2.0 The Concept Phase

The primary purpose of the **Concept Phase** is to determine whether an expert system is the appropriate technology for the application. In addition, it will be determined whether an expert system is a feasible project that is supported by the organization. For any given problem, there may be several technologies that could be used. By fully evaluating all viable paths, the team can ensure that the problem is not forced into an inappropriate technology. The **Concept Phase Checklist** (Fig. 2.1) provides a guide to lead you through the decisions that are made during the **Concept Phase**.

The idea for an expert system may be initiated by an expert, a developer of other expert systems, a potential user, or an outside proponent. Once the idea is presented to an organization, it must be evaluated to determine if an expert system is appropriate for the application. Although expert systems have been found to be a very useful technology for a wide variety of applications, they are not an appropriate solution for all problems. In some cases, a more conventional technology, such as spreadsheets or databases, are more suitable. In order to determine if an expert system is appropriate, the four major areas on the **Concept Phase Checklist** must be evaluated. These areas are:

- Applicability of the problem area to expert systems
- Benefits achieved by an expert system application
- Costs incurred by an expert system application
- Support for an expert system application.

As these areas are considered, they should be rated through the use of the **Concept Phase Checklist**. The criteria for determining the applicability of the problem area are fairly straightforward. The evaluation of the other areas, however, is more subjective, using a low - high rating scale. In order to rate these items effectively, carefully consider each aspect and use your best judgment to arrive at an accurate determination.

2.1 Is Expert Systems Technology Applicable?

Five key issues must be resolved in order to determine the applicability of the subject area to expert systems technology:

- Is the subject area well-defined?
- Is the process decision-based?
- Is an expert system the most appropriate technology?
- Is an expert available?
- Is there a need for knowledge distribution?

These issues are discussed below, and to assist you in the decision process, they have been broken down into their component parts in the **Concept Phase Worksheets** (Figs. 2.2 and 2.3).

CONCEPT PHASE CHECKLIST

APPLICABILITY OF PROBLEM AREA

			YES (1pt.)	NO (0pt.)	
C.1	SUBJECT AREA WELL DEFINED	≥5	✓		<5
C.2	DECISION-BASED PROCESS	≥3	✓		<3
C.3	EXPERT SYSTEM TECHNOLOGY MOST APPROPRIATE	≥4	✓		<4
C.4	EXPERT AVAILABLE	≥4	✓		<4
C.5	NEED FOR KNOWLEDGE DISTRIBUTION	≥4	✓		<4

Total: 5

BENEFITS TO BE ACHIEVED

	LOW 1	2	3	4	HIGH 5
IMPROVED PRODUCTIVITY				✓	
IMPROVED EFFICIENCY/TIME SAVINGS					✓
IMPROVED ACCURACY			✓		
IMPROVED CONSISTENCY			✓		
IMPROVED TRAINING				✓	
IMPROVED INFORMATION HANDLING			✓		
REDUCED DOWNTIME	✓				
OTHER ISSUES ______					

Total: 23

COSTS TO BE INCURRED

	LOW 5	4	3	2	HIGH 1
TIME AND LABOR				✓	
HARDWARE AND SOFTWARE		✓			
SYSTEM TESTING			✓		
SYSTEM DISTRIBUTION AND LICENSING		✓			
SYSTEM MAINTENANCE			✓		
UPGRADE MANAGEMENT AND VERSION CONTROL		✓			
OTHER ISSUES ______					

Total: 20

SUPPORT FOR EXPERT SYSTEM

	LOW 1	2	3	4	HIGH 5
DEVELOPERS					✓
END-USERS				✓	
MANAGEMENT				✓	

Total: 13

MILESTONE

		YES	NO	
WILL EXPERT SYSTEM TECHNOLOGY BE USED?	≥55	✓	☐	<55

Through the use of the supporting worksheets, ESTEL's subject area was found to be applicable to the use of an expert system. The design of telecommunications networks is an extremely complex and time consuming task, with the current process taking several months to complete. Significant time savings are expected through the use of an expert system, with the processing time being reduced to several days. Training time is also expected to be reduced dramatically. Costs are expected to be fairly high in the area of staff time and labor for the development, testing, and maintenance of the system, but the expected benefits greatly outweigh the production costs. Members of the organization, including experts, users, and management, are very supportive of the use of an expert system for this application.

Fig. 2.1 Sample Concept Phase Checklist

2.1.1 Well-Defined Subject Area

Before an expert system can be developed, the problem area must be identified. An expert system application requires a well-defined subject area, or domain. A well-defined domain must meet a number of criteria, as found in the **Concept Phase Worksheet**, section C.1 (Fig. 2.2):

- **Subject area contains structured data.** The subject area should be one that goes through a systematic process to solve a problem. Step-by-step decision processes are well-suited to expert systems.

- **Subject has clear boundaries.** The process must have definite starting and ending points. The most frequent problem in expert system development is selecting a large and unwieldy subject area that has no real starting and ending points. This will lead to frustration on the part of all involved, and potentially the demise of the project.

- **People are available who have knowledge in the subject area.** Expert systems are based on knowledge, so it is critical that there is someone who can provide that knowledge.

- **Subject area is at the correct stage of maturity.** A new subject that is unknown to the organization is not appropriate for an expert system. Neither is an area that is established to the point that a systematic process requiring little judgment has been developed, such as accounting or recordkeeping. The correct stage of maturity for an expert system application lies somewhere between, where information on the subject is available, but it is not common knowledge.

- **Domain of the expert system is narrow and isolated.** The subject area should be of a size that can be addressed by an expert system of approximately one hundred rules. A problem that can be solved by the expert over the phone would be a reasonable size for a small expert system. An isolated domain, or one with no outside effects or data requirements, is also well-suited to a small expert system.

- **Subject area is conducive to division into discrete segments or modules.** The subject area should be one that is easily divided into smaller segments. This will ease the development process, particularly in the areas of testing and modification.

- **Incremental progress is possible.** Related to the last issue, a subject area that can be segmented allows for several small expert systems to be developed incrementally and later linked together.

CONCEPT PHASE WORKSHEET

C.1 WELL-DEFINED SUBJECT AREA

	YES (1PT.)	NO (0PT.)
DATA IS STRUCTURED	✓	
BOUNDARIES ARE CLEAR	✓	
PEOPLE HAVE KNOWLEDGE IN AREA	✓	
KNOWLEDGE IS AT RIGHT STAGE OF MATURITY	✓	
SUBJECT IS NARROW & ISOLATED		✓
SUBJECT CAN BE DIVIDED INTO STAND-ALONE SEGMENTS	✓	
INCREMENTAL PROGRESS IS POSSIBLE	✓	

TOTAL: 6

C.2 DECISION-BASED PROCESS

	YES (1PT.)	NO (0PT.)
REQUIRES QUALITATIVE OR SUBJECTIVE REASONING	✓	
TASK IS COGNITIVE	✓	
TASK HAS MANY POSSIBILE COMBINATIONS	✓	
TASK INVOLVES CHAINS OF REASONING	✓	

TOTAL: 4

C.3 EXPERT SYSTEM TECHNOLOGY MOST APPROPRIATE

	YES (0PT.)	NO (1PT.)
QUANTITATIVE REASONING	✓	
DATA STORAGE & RETRIEVAL		✓
WORD PROCESSING		✓
MODELS & SIMULATIONS	✓	
PROCEDURAL PROGRAMMING		✓
RAPIDLY CHANGING DATA		✓

TOTAL: 4

The subject of telecommunications architecture design fits most of the criteria of a well-defined subject area. Although the subject is not narrowly bounded, it can be easily broken down into manageable segments. The domain meets all of the criteria for a decision-based process. The process does involve some quantitative reasoning and simulations, but these can be incorporated into outside programs.

Fig. 2.2 Sample Concept Phase Worksheet

This approach also facilitates system testing and maintenance after implementation.

If five or more of these criteria are met, then the subject area is well-defined (**Concept Phase Checklist**, C.1) and would be appropriate for an expert system application. If fewer than five of the criteria are met, then an expert system would not be appropriate.

2.1.2 Decision-Based Process

The domain of an expert system should be based on decisions, reasoning, or a logical process that can be duplicated through the use of rules or examples. Decision-based domains appropriate for expert systems are those that meet the following criteria found in the **Concept Phase Worksheet**, section C.2 (Fig. 2.2):

- **Require qualitative or subjective reasoning.** Expert system domains are generally based on judgmental decisions and heuristics, or rules of thumb. The subject should involve decisions that are based on experience rather than definitive facts.

- **Involve a cognitive task.** Cognitive tasks are those that you do in your head, rather than those involving complex computations or motor skills. These are generally qualitative or evaluative types of tasks that are well-suited to expert systems.

- **Involve many possible combinations.** Decision-based subjects generally involve many combinations and decision paths. As new problem elements are added, the number of steps required to reach a solution increase significantly.

- **Involve chains of reasoning.** Decision-based domains generally include topics that build on each other, and depend on each other for data. An example would be deciding what type of vehicle to drive. The selection depends in part on visibility. Visibility depends on the amount of illumination and weather conditions. Illumination depends on the time of day, availability of artificial light sources, and the presence of obstructions. Weather conditions depend on the time of year, time of day, and local factors.

If three or more of these criteria are met, then the domain is decision-based (**Concept Phase Checklist**, C.2), and would be appropriate for an expert system application. If fewer than three of the criteria are met, then an expert system would not be appropriate.

2.1.3 Expert System Technology Most Appropriate

The next issue to address is whether expert systems are the most appropriate technology to use for the application. There are several types of applications that are better suited to other conventional computing approaches. These include the following types of situations, found in the **Concept Phase Worksheet**, section C.3 (Fig. 2.2):

- **Tasks that involve quantitative reasoning**. Applications using extensive quantitative reasoning and calculations are usually better suited to a spreadsheet application.

- **Tasks focusing on data storage and retrieval.** Tasks requiring the manipulation of large amounts of data, with little reasoning about the data, are more effectively handled by a database system.

- **Applications involving word processing.** Word processing and frequent text changes are also inappropriate for an expert system.

- **Tasks requiring the use of models or simulations.** Specifically designed modeling and statistical packages are better suited to applications involving statistical modeling, simulations, and procedural processes.

- **Tasks requiring the use of procedural programming.** In a procedural program, the computer is told specifically how to perform a task, and there is no inferencing involved. Conventional programming languages are better suited to this type of application.

- **Subjects undergoing rapid changes**. An area that is rapidly changing in terms of the data or procedures, with new data becoming available frequently, is probably not appropriate for an expert system application. A rapidly changing domain of knowledge can result in the need for frequent upgrades to the expert system during the **Post-Development Phase** (see section 7.1). Furthermore, the system may be obsolete before it is even implemented.

If two or **LESS** of these characteristics apply (score >= 4), then expert systems technology is the most appropriate approach (**Concept Phase Checklist**, C.3). If four or **MORE** characteristics apply (score < 4), then expert systems technology is **NOT** the appropriate approach for the application.

2.1.4 Expert Available

A key element in the development of an expert system, as the name implies, is the expert. Thus, it is necessary to ensure that an expert or source of expertise is available before expert systems technology is selected. The following items, found in the **Concept Phase Worksheet**, section C.4 (Fig. 2.3), must be confirmed to determine that an expert is available :

- **An expert exists.** First, it must be determined that a true expert exists in the subject area. An expert is someone who is:

    - Experienced in the subject area
    - Respected and consulted by colleagues
    - Better than novices at the task.

- **A specific expert can be identified.** An individual can be identified who has sufficient experience to provide the basis of knowledge for an expert system.

- **Expert is willing to be involved.** The existence of an expert is not enough. The expert must be willing to be involved in the project and see it through to completion. An unwilling expert can withhold data, be uncooperative, and jeopardize the project.

- **Expert is available and accessible.** An expert must also have the time available to commit to the project and be accessible to the knowledge engineer. Prior to the beginning of development, the knowledge engineer should draft a schedule of the knowledge acquisition process and subsequent testing and refinement with the expert to identify availability and timing issues.

- **Expert can communicate detailed knowledge.** The expert must also be able to convey the knowledge to the knowledge engineer. The most knowledgeable people on a subject will be of little or no value if their knowledge cannot be captured.

If four or more of these items are true, then an expert is available (**Concept Phase Checklist, C.4**), and an expert system could be developed. If fewer than four items are true, then an expert system would not be appropriate for the application.

2.1.5 Need for Knowledge Distribution

Expert systems are well suited to applications where there is a need for the expertise to be distributed to other users. The domain of an expert system is frequently a complex step-by-step process which is difficult to learn, but through an expert system, the procedure can be applied by

CONCEPT PHASE WORKSHEET (cont.)

C.4 EXPERT AVAILABLE

	YES (1PT.)	NO (0PT.)
EXPERT EXISTS (BETTER THAN AMATEURS)	✓	
EXPERT CAN BE IDENTIFIED	✓	
EXPERT IS WILLING OR EAGER	✓	
EXPERT IS AVAILABLE AND ACCESSIBLE	✓	
EXPERT CAN COMMUNICATE DETAILED KNOWLEDGE		✓

EXPERT'S NAME Bob Smith

TOTAL: 4

C.5 NEED FOR KNOWLEDGE DISTRIBUTION

	YES (1PT.)	NO (0PT.)
OTHERS COULD BENEFIT FROM KNOWLEDGE	✓	
PROCESS WOULD BE IMPROVED	✓	
NEED FOR KNOWLEDGE TRANSFER		✓
PROBLEM WITH EXPERT ACCESSIBILITY		✓
NEED FOR TRAINING	✓	
GEOGRAPHIC DISTRIBUTION PROBLEM		✓
IMPROVEMENTS NEEDED IN DECISION-MAKING PROCESS	✓	

TOTAL: 4

Several experts were available on the subject of network design. An expert named Bob Smith was selected to provide the basis for ESTEL's knowledge base. Although he sometimes has trouble communicating his ideas, he has a great deal of experience in the subject area. He is also very interested in the project, and is available to work on it. A need for knowledge distribution was found in the organization, mainly in the areas of improving the process and providing training to new staff members.

Fig. 2.3 Sample Concept Phase Worksheet (cont.)

a large number of people. Common reasons for knowledge to be distributed include the following (Fig. 2.3, section C.5):

- **Others could benefit from the knowledge**. An understanding or working knowledge of the subject area could improve the morale or efficiency of the organization.

- **Process would be improved**. The use of an expert system would improve the efficiency or consistency of a process.

- **Need for knowledge transfer**. There is a need to document the institutional memory of someone who is leaving the organization. In this way, the individual's experience can still be utilized by the organization.

- **Problem with expert accessibility**. There is a need to capture the knowledge of an expert who is often unavailable.

- **Need for training**. The organization has a high turnover rate or a rapidly growing staff of new personnel who need to be trained.

- **Geographic distribution problem**. Knowledge is needed at a large number of widely spread locations, requiring the expert to travel extensively.

- **Decision-making process would be improved**. An expert system would relieve the burden of decision making from a single person, and allow the expert to concentrate on the most complex problems.

If four or more of these items can be identified, then there is a need for knowledge distribution, which could be addressed by an expert system (**Concept Phase Checklist**, C.5). If fewer than four of the items are identified, then expert systems may not be the appropriate technology to use for the application.

After items C.1 - C.5 have been completed on the **Concept Phase Worksheets**, the results are entered in the first section of the **Concept Phase Checklist**. In order for expert systems to be applicable to the subject area, at least four of items C.1 - C.5 should be answered with a "YES" on the **Concept Phase Checklist**. If more than one of these items receives a "NO" response, then expert systems are not appropriate for the application.

2.2 Benefits to be Achieved

Also critical to the **Concept Phase** is determining whether an expert system is a cost-effective solution to the problem. The economic feasibility of an expert system can be determined by comparing the estimated benefits and costs of expert system development. The use of an expert system can result in benefits in a number of areas. To determine if an expert system would

benefit a given area, pose the following questions and indicate the level of benefits expected on the **Concept Phase Checklist** (Fig. 2.1):

- **Productivity** - Is the process boring or monotonous? Is there a bottleneck in the information flow of the organization?

- **Efficiency and time savings** - Is the process time-consuming? Does it require several people?

- **Accuracy** - Are the results inaccurate and prone to human error?

- **Consistency** - Are the results inconsistent? Do the results vary by person or location?

- **Tr. 'ning** - Is a lot of time and staff required for training? Are experienced personnel unwilling to let novices solve problems?

- **Information handling** - Is the process confusing, or does it involve several sources of information, resulting in time delays?

- **Reduced downtime** - Is there excessive downtime that could be lessened through improved decision-making?

- **Other issues** - Are there any other areas that would benefit from the use of an expert system?

These criteria should be rated based on the improvements that are expected through the implementation of an expert system. Areas where there is currently a problem or bottleneck that could be corrected through the use of an expert system should be given a high rating. Areas that are not a problem or would not be improved significantly by an expert system should be given a lower rating.

The desired benefits of the expert system development effort will be incorporated into the **Requirements Statement** during the **Definition Phase**, then later used during the **Test and Evaluation Phase** to measure its economic effectiveness, and the organizational impact of the expert system, as shown in the **MAU Framework** (Appendix A).

2.3 Costs to be Incurred

The benefits that will result from a specific expert system application should be assessed and compared with the potential costs of developing and implementing an expert system. The level of expected costs should be evaluated on the **Concept Phase Checklist** (Fig. 2.1). Common development costs include:

- **Time and labor** - the development process may require a significant number of hours from several staff members, including the expert, developer, and manager.

- **Hardware and software** - purchasing and licensing fees must be considered if new hardware or software is required.

- **System testing** - the time of several personnel will be required for system testing.

- **System distribution** - the distribution of the system may require the purchase of disks and packaging equipment, as well as the time of a staff member to prepare the expert system for distribution and maintain information on the users.

- **System maintenance** - a staff member will be needed to maintain the system, perform periodic reviews of the system, and provide technical support.

- **Upgrade management and version control** - a staff member will be responsible for keeping track of software upgrades and deciding when an upgrade of the system is necessary. There may also be a charge for software upgrades.

- **Other issues** - there may be other costs that are specific to the organization. These should be noted here.

These criteria should be rated based on the expenses that can be expected with the implementation of an expert system. If the expert system would require investments in new equipment or intensive labor for development and implementation, then these areas should be given a high rating. If minimal expenses will be incurred, low ratings should be given.

The proposed costs of the expert system application will be recorded in the **Requirements Statement** during the **Definition Phase**. Cost and benefit issues should be kept in mind throughout the development process, and will be reassessed during the **Test and Evaluation Phase** to determine the actual impacts of developing the expert system.

2.4 Support for an Expert System

The final component of the **Concept Phase** is to ensure support for the project. One of the key factors to consider in determining if an expert system should be developed is whether the project is important to the organization. Without adequate support, an expert system is likely to fail, because few resources will be committed to the project and few significant benefits will be achieved by the organization. When an expert system fails, it also jeopardizes the chances for future expert system to be developed. Developing an expert system requires the time, effort, and cooperation of everyone involved. For the project to be successful, it is important that all affected parties - the developers, expert, management, and users - are aware of the goals, objectives, and requirements of the expert system. The presence of a proponent - a staunch supporter of the project, who is either integrally involved in the development, or outside of the process - can also be a critical factor to the success of the project. Such an individual could

ensure that development continues to move forward, even if interest wanes among other members of the team.

To determine the level of support for an expert system in the organization, an informal survey should be conducted on the staff members, to determine the amount of time and energy they are willing to commit to the project. Their responses should be rated accordingly. The results of this survey should be recorded on the **Concept Phase Checklist** (Fig. 2.1).

2.5 Summary

By the end of the **Concept Phase**, the use of an expert system will be fully evaluated for its applicability, expected benefits and costs, and organizational support. In order to complete the **Concept Phase** milestone, the decision is made as to whether expert systems technology is appropriate for the problem at hand. If at least four of the items C.1 - C.5 are "YES" and the totals on the **Concept Phase Checklist** add up to 55 or more, then expert systems technology should be used for the application. If the total is less than 55 and an expert system is not found to be an appropriate tool for the job, then another technology should be considered. The evaluation process should provide a basis for selecting another more suitable technology. If an expert system is selected, proceed to the **Definition Phase**.

3.0 The Definition Phase

Once a decision has been made to go forward with an expert system application, the **Definition Phase** begins. It is during this phase that most of the key decisions in the project will be made. The problem area will be clearly defined, the team members selected, and the development processes and schedule documented. Actions taken at this time will have a significant effect on the success of the project, and accordingly, should be given serious consideration. The **Definition Phase Checklist** (Fig. 3.1) provides a guide for completing the tasks involved in the **Definition Phase**.

3.1 Defining the Scope and Requirements

One of the first and most important steps to be taken after expert system technology is selected, is to fully define the expert system. This will include addressing the issues of:

- **Scope** - What will the expert system cover?
- **Requirements** - Where will it operate, what features and performance are necessary?
- **Purpose** - Why is it being developed?
- **Intended Users** - Who will use it?
- **Capabilities** - What can it do?
- **Limitations** - What can it not do?
- **Outputs** - What solutions will it produce?
- **Expectations** - What will the benefits be?
- **Maintenance** - How will it be supported?

Careful definition of the expert system is essential. The developers, users, and management should have a thorough understanding of the application, including what the system will and will not do, key assumptions and limitations, system outputs, and conditions of operation. Issues such as response time, level of accuracy, screen formats, and reports should be presented in sufficient detail. During the **Definition Phase**, the knowledge engineer should determine the users' level of expertise with computers and their familiarity with the specific subject area. This information will help the developers make the software understandable to the users, but not overly simplistic.

3.1.1 Requirements Statement

Each of the issues in the definition of the expert system will be incorporated into a document known as the **Requirements Statement** (Fig. 3.2), the first item on the **Definition Phase Checklist**. This is a flexible, working document that will later become a part of the system documentation. The **Requirements Statement** need not be a lengthy document; it may be only one to two pages in length, as long as it covers all of the relevant issues (see Fig. 3.2). It will be reassessed periodically during the development phases and revised when necessary. As development progresses, issues will surface that were not apparent when the system was originally defined. These issues will be reviewed as they arise, and the **Requirements Statement** will be

DEFINITION PHASE CHECKLIST

SCOPE AND REQUIREMENTS DEFINED*

		DATE
D.1 REQUIREMENTS STATEMENT COMPLETED	✓	6/8/90

DESIGN OF EXPERT SYSTEM PREPARED*

D.2 ONE-PAGE DESIGN COMPLETED	✓	6/11/90
D.3 DETAILED DESIGN COMPLETED	✓	6/12/90

DEVELOPMENT TEAM SELECTION*

EXPERT Bob Smith

DEVELOPER(S) George Martin

MANAGER Mary Williams

OUTSIDE REVIEWER (IF APPLICABLE) ____________

KNOWLEDGE ACQUISITION APPROACH SELECTION

(INDICATE DEGREE OF USE)

ELICITING KNOWLEDGE	LOW				HIGH
STRUCTURED INTERVIEWS				✓	
UNSTRUCTURED INTERVIEWS			✓		
QUESTIONNAIRES	✓				
OBSERVATION		✓			
DOCUMENTS			✓		
DOCUMENTING KNOWLEDGE					
DECISION TREES				✓	
TABLES				✓	
KNOWLEDGE MAPS	✓				

END USER CONTACT	LOW				HIGH
LEVEL OF INTEREST				✓	
UNDERSTANDING OF SUBJECT AREA					✓
LEVEL OF COMPUTER EXPERTISE			✓		

COMMENTS Expert is one of the more experienced users.

* MILESTONES

During the **Definition Phase**, the **Requirements Statement**, the **One-Page Design** and the **Detailed Design** for ESTEL were completed. The development team was also selected. The main method selected for eliciting knowledge was the use of interviews, with documents and observation being used during the domain orientation period. Decision trees and tables were selected as the means for documenting the knowledge. The users in this case are experts on the subject of network design. Thus, they were found to have a high level of understanding in the subject area, and were also very interested in the use of an expert system.

Fig. 3.1 Sample Definition Phase Checklist

D.1 REQUIREMENTS STATEMENT

SUBJECT AREA Telecommunications Network Design

SCOPE (SPECIFIC BOUNDARIES WITHIN SUBJECT AREA)

Physical protection, electronic protection, and route & media diversity for entire network at an overview level of detail.

SYSTEM REQUIREMENTS (HARDWARE, SOFTWARE, ETC.)

PC with 80286 chip, 640 K RAM, 5 mg. hard disk space, EGA graphics, DOS 3.0, expert system shell software.

PURPOSE (ROLE IN PROCESS, EXPECTED RESULTS, ETC.)

Follows several network analysis models. Suggests enhancements to the networks. Presents results to users for review. Produce recommendations within 2-3 days, improve productivity, accuracy, consistency.

INTENDED USERS (WHO WILL USE IT)

Telecommunications network designers that currently analyze model results manually and make improvements incrementally.

CAPABILITIES (WHAT IT WILL DO)

Examines each node of the network and recommends necessary improvements. Provides justifications and refines model outputs.

LIMITATIONS (WHAT IT WILL NOT DO)

Does not optimize costs across network or examine detailed threat information. Will require expert review. No specific equipment.

OUTPUTS (FILE INTERACTIONS, DEPENDENCIES, REPORTS, ETC.)

Design alternatives for each node of the network and survivability ratings.

EXPECTATIONS (BENEFITS, IMPROVEMENTS, ETC.)

Decrease time to design network by several weeks. Improve productivity of each team member. Help train new designers.

ROUTINE MAINTENANCE AREAS

Model interfaces, improvements or policy changes in telecommunications network design procedures.

Fig. 3.2 Sample Requirements Statement

updated to adjust to the changing situation. When the initial **Requirements Statement** is completed, check this item off on the **Definition Phase Checklist** (Fig. 3.1) and indicate the date of completion.

The developers, users, and management should all be involved in the preparation and revision of the **Requirements Statement** to ensure that the project remains on track and within resource constraints. Its main purpose is to serve as a guideline for expert system development and provide a means for everyone involved to work toward the same goal.

This document will also be particularly useful during the **Test and Evaluation Phase**. By using the **Requirements Statement**, the testing team can determine if the expert system accomplishes what was intended. A successful expert system application is one that effectively and accurately performs according to its design. Any goals that have not been achieved will be addressed during the **Final Development Phase**, and either incorporated into the expert system or removed from the **Requirements Statement**.

3.2 System Design

In addition to identifying and refining the requirements of the expert system, it is also important to carefully design the expert system during the **Definition Phase**. Developing an expert system design, the second item on the **Definition Phase Checklist** (Fig. 3.1), not only provides a blueprint for the developers to follow, it also encourages the developers to think through the entire process and identify all interfaces and interdependencies of the system.

An expert system design is often developed in the form of two documents - the **One-Page Design**, and the **Detailed Design**. These two documents should be developed as fully as possible during the **Definition Phase**. They will be reviewed and revised throughout the development process as more information becomes available to the developers. These documents will also be re-evaluated during the **Prototype Phase** and the **Test and Evaluation Phase** to ensure that development is progressing according to the design.

The **One-Page Design** (Fig. 3.3) describes all aspects of the expert system at a high level. Components of the **One-Page Design** include:

- System overview
- Listing of system components
- System structure
- Interfaces
- External calls and data sources
- Knowledge representation
- Inferencing mechanism.

The **Detailed Design** (Fig. 3.4) goes into more detail, breaking down the system into its component parts and describing each fully. The **Detailed Design** should include:

- A breakdown of each component part
- Purpose of each component

D.2 ONE-PAGE DESIGN

SYSTEM OVERVIEW: Run through network data in batch mode to determine results. Interact with user through justification rules.

SYSTEM COMPONENTS

Initialization and finalizing rules, recommendation rules, control rules, justification rules

BASIC SYSTEM STRUCTURE (ATTACH APPLICABLE DRAWINGS)

Initialize data and files. For each node, check physical and electronic protect and rate media diversity for suggested enhancements. Write results to output files.

SYSTEM INTERFACES

Data files from models, user interaction via interface

EXTERNAL CALLS AND DATA SOURCES

No calls to external programs. 4 data files consisting of node and link data and results from routing and threat simulations.

KNOWLEDGE REPRESENTATION

Production Rules

INFERENCING MECHANISM

Forward Chaining

Fig. 3.3 Sample One-Page Design

D.3 DETAILED DESIGN

COMPLETE THIS FORM FOR EACH SYSTEM COMPONENT

COMPONENT: Physical Protection Rules

PURPOSE: To recommend desired level of physical protection for each node in the network.

SCOPE: Uses threat, node, and link data to recommend level of hardening: minimal, light, moderate, or heavy. Does not include specific equipment or cost information.

NECESSARY INFORMATION: Node's current hardening status, colocation information and importance.

SPECIFIC EXTERNAL CALLS: None

SPECIFIC DATA SOURCES: Node, link, threat file data

Fig. 3.4 Sample Detailed Design

- Scope of each component
- Necessary information
- Specific external calls
- Specific data sources.

Upon completion of these two design documents, check them off on the **Definition Phase Checklist** and indicate the dates of completion.

3.3 Development Team

Before the development of an expert system begins, a team is selected. The most effective team will consist of people who are familiar with or willing to learn the expert system development process, and are dedicated to producing a successful application. The number of members on an expert system development team can vary based on the size of the application and the experience of the team members. In the case of a small expert system application, the team will consist of one to three members who will perform three main roles:

- Expert
- Developer
- Manager.

The users should also be considered members of the team, as they will be closely involved in the development process, as well as the testing and final acceptance of the system.

In the development of a small expert system, one person is often required to fill multiple roles. Because expert system shells are widely used, it is common for a single person, who we will refer to as the developer, to serve as both the knowledge engineer and the programmer. These two roles fit well together because the knowledge engineer knows how to best record and document the knowledge for later input into the expert system shell. While programming, this individual can also locate gaps or contradictions in the knowledge and return this information to the expert, in the role of the knowledge engineer, for clarification.

By combining the development roles, an expert system application can be completed more efficiently, and by a smaller team. However, it is important to keep in mind the drawbacks of a small development team. For example, if one person serves as both the expert and the knowledge engineer, there is a greater likelihood for the system to be biased, or for information to be left out. Alternatively, if the knowledge engineer is also a user, the system may be biased toward this person's specific preferences, while things that might be useful to others are left out. In cases such as these, where the system of checks and balances provided by a separate individual for each role is not available, the developers should keep this in mind and arrange for at least one outside party, such as an independent tester, to review the system.

As the development team members are selected, enter their names on the **Definition Phase Checklist** (Fig. 3.1). The selection of a development team is a milestone for this phase of the expert system development process.

3.3.1 Expert

The first role to be filled is that of the expert, or the source of knowledge. There may be more than one expert, who:

- Has experience in the subject area
- Is respected and consulted by colleagues
- Has the ability to communicate
- Has time to commit to the project
- Has a desire to share expertise.

If the source of knowledge is a document - such as a set of regulations or a training manual - there may be no expert.

3.3.2 Developer

Another key role on the expert system development team is the developer, who serves as both the knowledge engineer and the programmer. In the role of the knowledge engineer, the developer is instrumental in a number of areas, including:

- Refining the problem area
- Designing the expert system
- Acquiring knowledge
- Selecting a knowledge representation
- Encoding the knowledge
- Implementing, testing, and evaluating the expert system
- Contacting users for their input.

Also serving in the role of the programmer, this individual should be familiar with computers, and be willing to learn an expert system development shell. As the programmer, the developer is responsible for building the expert system in an expert system shell, using the knowledge that was acquired during the knowledge acquisition sessions, incorporating revisions and changes, and participating in the testing.

The developer will also be responsible for several other tasks throughout the development process, including:

- Designing the user interface
- Writing the documentation
- Verifying the expert system
- Providing training and technical support
- Maintaining the system.

3.3.3 Manager

The final role to be filled is the manager or project proponent. This individual should possess the following qualities:

- An understanding of the expert system development process
- Good arbitration skills
- The ability to motivate people to work together
- Support for the project and a desire for a successful outcome.

The manager should be involved in all phases of the development process, and should communicate frequently with the team members to ensure that progress is being made. The manager will be responsible for making numerous decisions throughout the development process and ensuring that the schedule and budget are consistent with the project objectives.

3.4 Knowledge Acquisition Approach

An early responsibility of the knowledge engineer is the selection of a knowledge acquisition approach. Knowledge acquisition is the process by which the knowledge engineer extracts knowledge from the expert. Because this knowledge will serve as the basis for the expert system, it is imperative that the knowledge engineer is thorough and effective in eliciting information from the expert. In selecting a knowledge acquisition approach, the knowledge engineer must decide on methods to accomplish two major tasks:

- Elicit knowledge
- Document knowledge.

3.4.1 Methods of Eliciting Knowledge

In the case of one or more human experts, the knowledge is generally elicited through a series of interviews between the expert and the knowledge engineer. There are several effective knowledge acquisition interviewing methods, including:

- Structured interviews
- Unstructured interviews
- Observation.

The knowledge engineer should select one or more methods for interviewing based on the application requirements and the degree of precision required in the final system. Several interviewing techniques and other knowledge acquisition methods, as well as their advantages and disadvantages, are described in Fig. 3.5. The knowledge engineer should become generally familiar with the application area before interviewing the expert. This facilitates effective discussion because common terms of the trade can be used to identify key concepts and to save time.

TECHNIQUE	USES	ADVANTAGES	DISADVANTAGES
UNSTRUCTURED INTERVIEW	PROVIDE BACKGROUND, INTRODUCTION, FAMILIARIZATION, EARLY IN PROCESS	ALLOWS EXPERT TO EXPOUND, BUILDS RAPPORT, IDENTIFIES IMPORTANT AREAS	NO FOCUS, TIME CONSUMING, CAN RESULT IN INCOMPLETE KNOWLEDGE
STRUCTURED INTERVIEW	GATHER SPECIFIC INFORMATION, ELABORATE ON DETAILED PROCESSES	FOCUSES ON PRECISE KNOWLEDGE, FOLLOWS ALL OUTPUT PATHS IN A STEP-WISE MANNER	MAY NOT FOLLOW EXPERT'S WAY OF THINKING, MAY BE TOO RIGID AND DISCIPLINED
TWO-ON-ONE INTERVIEW	PROVIDE FOR MORE EFFICIENT DATA COLLECTION	ALLOWS FOR MORE INFORMATION TO BE OBTAINED AND MORE ACCURATE RECORDING	CAN INTIMIDATE EXPERT, CAN ALSO BE VERY TIRING TO EXPERT
QUESTIONNAIRES	PROVIDE GENERAL INFORMATION ON BASIC QUESTIONS	FAST AND NON-THREATENING, GOOD IN CASES WHERE EXPERT IS FAR AWAY	VERY LIMITED IN THE KNOWLEDGE THAT IS OBTAINED, DOES NOT ALLOW FOR EXPANSION
OBSERVATION	UNDERSTAND ACTUAL DECISION-MAKING PROCESS	SHOWS HOW EXPERT THINKS AND SOLVES ACTUAL PROBLEMS	CANNOT OBSERVE ENTIRE THOUGHT PROCESS, IMPORTANT INFERENCES MAY BE MISSED

Fig. 3.5 Methods of Eliciting Knowledge

In some cases, the source of knowledge is a document or set of documents, instead of a human expert. The knowledge acquisition process must be adapted to accommodate these conditions. Instead of performing interviews, the knowledge engineer will read the documents to acquire knowledge. Although there may not be an expert on the subject, there is probably someone who is familiar with the documents, and whose assistance would be useful to the knowledge engineer. This process takes a little longer because the knowledge engineer does not have an expert to answer questions, explain key concepts, important factors, and data inter-relationships.

The selection of a knowledge acquisition approach is addressed in the **Definition Phase Checklist**. For each approach, indicate the degree to which each of these knowledge elicitation methods will be used. A good knowledge acquisition approach will incorporate many or all of these methods, but the use of interviews should be emphasized. An expert system that is based mainly on questionnaires and documentation will not be as good as one that is based on an expert's experience.

3.4.2 Methods of Documenting Knowledge

Next, the knowledge engineer determines an effective method to document the knowledge that is acquired during the knowledge acquisition sessions. Common methods of documentation include converting the knowledge into:

- Decision trees
- Decision tables
- Knowledge maps.

See Figs. 4.8 - 4.10 for examples of these graphical representations of the knowledge. Indicate on the **Definition Phase Checklist** (Fig. 3.1) the degree to which each of these documentation methods will be used.

3.4.3 Knowledge Testing Issues

When planning the knowledge acquisition process, it is important for the knowledge engineer to keep in mind several relevant factors that apply to testing the validity of the knowledge. First, the knowledge engineer should identify probable areas for gaps or omissions in the knowledge. It often happens that the knowledge provided by the expert, while correct, is incomplete. The knowledge engineer should also plan to set aside a number of test cases that will be used during the **Test and Evaluation Phase** for the verification of the expert system. By keeping in mind these testing issues, the developer will have a headstart on the testing process, and will be better prepared for the later testing of the knowledge base, as shown in the **MAU Framework** (Appendix A).

3.5 User Contact

Before the **Prototype Development Phase** begins, the knowledge engineer and manager should meet with the users. The purpose of the expert system application, as well as the overall expert system development process should be reviewed at this time. The users should also be given the opportunity to voice their expectations and concerns about the system, in order for management to address these issues early in the development process. Using the **Definition Phase Checklist**, the knowledge engineer should evaluate the user group for their:

- Level of interest
- Understanding of the subject area
- Level of computer expertise.

After surveying several users on these issues, an average of the responses should be recorded on the **Definition Phase Checklist** (Fig. 3.1). If the ratings are low in each of these areas, then the response to and use of the system will most likely be low. If this is the case, it may be necessary to reconsider the use of an expert system for the problem, or to begin educating the users so they are prepared for the expert system when it becomes available.

This information will also be helpful in setting up the user interface, and determining the focus of the system. At this time, the knowledge engineer can also identify individuals to participate in the user testing portion of the **Test and Evaluation Phase**.

3.6 Summary

By the end of the **Definition Phase**, the expert system application is fully defined, the people involved in the development process have been introduced to the project, and preparations have been made that will be implemented in the remaining development phases. The following three milestones will also be completed:

- **Requirements Statement** produced
- Design documents completed
- Development team selected.

These milestones are attained throughout the phase, as each of these items is checked off on the **Definition Phase Checklist**.

4.0 The Prototype Development Phase

The **Prototype Development Phase** implements the results of the **Definition Phase** in the development of a prototype expert system. A prototype expert system is an initial draft of an expert system, in which the knowledge and processes can be tested and presented to the users, then later expanded or improved, if necessary. Though described sequentially, much of the **Prototype Development Phase** is iterative, with successive passes incrementally adding to and refining the expert system. Rapid prototyping techniques are used in the development of expert systems, as well as conventional computer systems, to produce a working system quickly. This technique also helps to solidify the requirements by showing the users at a very early stage what they will be getting in the finished application.

The **Prototype Development Phase** contains several steps, each of which is present in some form in every expert system project - regardless of size and complexity. These steps, which can be traced in the **Prototype Development Phase Checklist** (Fig. 4.1), are described in further detail below, and include:

- Selecting a development shell
- Consulting with users on user interface issues
- Completing domain orientation
- Conducting knowledge acquisition sessions
- Identifying test cases for later use
- Converting knowledge into the selected knowledge representation
- Performing incremental testing of the system
- Reviewing **Requirements Statement** and design documents
- Preparing documentation for the prototype
- Iteratively developing a full prototype system.

As stated previously, it is very likely that many of the items on the list will be performed through several iterations, each improving on the previous effort. After several preliminary evaluations, the system requirements will begin to stabilize.

4.1 Selecting an Appropriate Development Shell

It is assumed that the expert system application being considered will be developed using an expert system shell. Expert system shells are very useful because a non-programmer can quickly learn the software and develop a small expert system without extensive training. Advantages of using an expert system shell include reduced development time, ease of use, and a tested development and delivery environment. Disadvantages include restrictions on user and developer interfaces, limitations on knowledge representations and inferencing mechanisms, and constraints on integration with external files and programs.

PROTOTYPE DEVELOPMENT PHASE CHECKLIST

DEVELOPMENT TASKS

			DATE(S)
P.1	EXPERT SYSTEM SHELL SELECTED	✔	6/18/90
P.2	USERS CONSULTED ON INTERFACE ISSUES	✔	6/20/90
P.3	DOMAIN ORIENTATION COMPLETED	✔	6/25/90
P.4	KNOWLEDGE ACQUISITION SESSIONS COMPLETED	✔	7/13/90
P.5	CASES SET ASIDE FOR TESTING	✔	7/13/90
	KNOWLEDGE CONVERTED INTO KNOWLEDGE REPRESENTATION	✔	7/25/90
	INCREMENTAL TESTING PERFORMED	✔	7/27/90
P.6	REQUIREMENTS STATEMENT REVIEWED	✔	8/1/90
P.7	DESIGN DOCUMENTS REVIEWED	✔	8/1/90
P.8	DOCUMENTATION PREPARED	✔	8/1/90

MILESTONE

PROTOTYPE EXPERT SYSTEM PRODUCED	✔	8/1/90

Fig. 4.1 Sample Prototype Development Phase Checklist

To select an expert system shell, important evaluation criteria are first identified and then ranked, using section P.1 in the **Prototype Phase Worksheet** (Fig. 4.2). Selection criteria for choosing a specific shell include:

- Availability in the organization
- Knowledge representation
- Inferencing mechanism
- User and developer interface
- External data and program interface
- Development and delivery software costs
- Licensing issues
- Hardware requirements
- Maintenance issues.

For each of the criteria that apply, a rank should be assigned, in order of importance to the organization. For example, if five of these items are found to be important, they should be ranked from 1 to 5, with 1 being the most important and 5 the least important. These criteria will later be used to develop a shell evaluation matrix. After the shell selection criteria have been considered and are ranked in order of importance, the **Expert System Shell Evaluation Matrix** (Fig. 4.3) should be completed. Fill in the most important criteria from P.1 across the top of the matrix, and the shells being evaluated along the left side, then check off the criteria that are met by each shell. Select the shell that best meets the criteria.

The following subsections describe in more detail a number of the expert system shell evaluation criteria that may be important to an organization.

4.1.1 Current Availability

The current availability of a shell in the organization indicates that someone is already familiar with the product from a development and/or user standpoint. This existing knowledge of the software will be beneficial for several reasons. First, the development time will be reduced if the developers do not have to learn how to use the shell. It will also be easier to discuss the system with the users if they are already familiar with the software. The overall development costs may be reduced if the development version of the software is already available. The licensing and distribution fees may also be taken care of. Possibly the most important factor is that the presence of the shell in the organization indicates an interest in expert systems, and improves the chances for a successful system.

4.1.2 Knowledge Representation

The type of knowledge representation available in an expert system shell may or may not be an important selection factor. In some cases, the type of expert system or the nature of the knowledge may lend itself to one type of representation over another. The most common knowledge representations used in expert system development shells are:

- Rules
- Examples.

PROTOTYPE PHASE WORKSHEET

P.1 EXPERT SYSTEM SHELL SELECTION

SELECTION CRITERIA:	RANK
- AVAILABILITY TO ORGANIZATION	2
- KNOWLEDGE REPRESENTATION	4
- INFERENCE ENGINE	5
- USER AND DEVELOPER INTERFACE	3
- INTEGRATION WITH OTHER PROGRAMS/FILES	1
COST	___
EASE OF DISTRIBUTION	___
HARDWARE REQUIREMENTS	___
MAINTENANCE ISSUES	___
OTHERS: ______________	___
______________	___

P.1.1 EVALUATION MATRIX COMPLETED ☑

SHELL SELECTED: CLIPS

P.2 USERS CONSULTED ON INTERFACE ISSUES

DESIRABLE FEATURES:	RANK
- MENUS	4
- GRAPHICS	6
FUNCTION KEYS	___
- REPORTS	3
- FILE MANAGEMENT	1
- HELP FACILITIES	5
SIMILARITY TO CURRENT SYSTEMS	___
- JUSTIFICATION CAPABILITIES	2
OTHERS: ______________	___
______________	___
______________	___

Five expert system shell criteria were found to be important by the developers of ESTEL. These five items were ranked in order of importance. A shell selection matrix was then developed, and the most appropriate shell was identified. Next, users were consulted on interface issues, and the features most important to them were identified and ranked.

Fig. 4.2 Sample Prototype Phase Worksheet

P.1.1 EXPERT SYSTEM SHELL EVALUATION MATRIX

CRITERIA / SHELL	Integration with external files	Available in Organization	User and Developer Interface Acceptable	Appropriate Knowledge Representation	Suitable Inference Engine
EXSYS	✓	✓		✓	
CLIPS	✓	✓	✓	✓	✓
1st Class	✓				✓
Level 5	✓		✓	✓	✓
KBMS	✓		✓	✓	✓

Fig. 4.3 Sample Expert System Shell Evaluation Matrix

The most prevalent knowledge representation approach for small expert systems, and the one most commonly used by expert system shells, is rules. This approach is preferred because rules are easy to understand, logically consistent, and often straight-forward to test. Rules are most applicable to structured subject areas that step through a number of decision points.

The selection of a shell based on the knowledge representation is more likely to become an issue if an example-based representation is desired, since fewer shells have this capability. Examples can be useful when the problem is not particularly well understood and sufficient quantities of data and case histories are available to use as example cases. The decision produced by this induction method often requires modification, but it provides a good initial cut at the problem. In addition, examples can be helpful in refining requirements that may be implemented later as rules.

4.1.3 Inferencing Mechanism

The selection of an inferencing mechanism is generally not a key issue in the selection of an expert system shell. An inferencing mechanism uses the information stored in the knowledge representation to make decisions and derive conclusions. The primary types of inferencing mechanisms available in expert systems shells are forward and backward chaining.

Forward chaining is a data or event driven approach to exploiting rules. This method deduces its conclusions from the data available. Forward chaining is most commonly used for applications with a vast number of potential solutions, including design, configuration, and some types of planning applications.

Backward chaining hypothesizes solutions and attempts to prove or disprove them. Backward chaining is frequently chosen for classification types of applications with a limited number of potential solutions, including diagnostics, monitoring, and repair. Backward chaining is an interactive form of inferencing that focuses on relevant portions of information and thus provides intelligent user questioning.

Most shells have the capability of processing the logic without the developer having to worry about which mechanism is being used. The inferencing mechanism could become an issue if a larger system is developed and efficiency and response time are critical.

4.1.4 User/Developer Interface

An expert system shell's interface is often a major selection criteria. Since many of the shells are similar in other respects, an easy to use interface that is enjoyable to work with and meets the needs of the users and developers is often an important aspect of the software. Important features to look for may include:

- Menus
- Function keys
- Graphics
- Flexibility
- Ease of modification
- Ease of learning
- Similarity to existing software.

4.1.5 Program Interface

Some applications require the expert system to interface with external data files or programs. When this is the case, it is important to ensure that the shell is compatible with the software that it will be required to interact with. Also important are factors such as:

- Memory requirements
- System call capabilities
- Disk space
- Operating system version requirements.

4.1.6 Cost

The cost of an expert system shell can sometimes be a deciding factor to an organization. Shells are available in a wide range of prices, from the low hundreds to thousands of dollars. The available features and functions are generally associated with the cost of the software. Thus, if cost is a key issue, the price range should be set before some of the other options, such as interfaces, are evaluated.

4.1.7 Distribution and Licensing

Another important issue to consider is the distribution and licensing of the application. Most expert system shells are available in two versions - a development version and a run-only, or delivery, version. The development version is the full system in which the knowledge base, knowledge representations, inferencing mechanisms, and interface are developed, and internal testing is performed. This is generally a relatively costly piece of software that is only needed by the developers and maintenance personnel. Frequently, the users of an expert system only need to be able to run the application. For this reason, most expert system shells are available in a run-only version. This is much smaller and less costly than the development version, and is more appropriate for wide distribution of the application.

The issues that surround the licensing costs of expert system software result from the various approaches that software companies take regarding run-only versions. In some cases, the development version includes a run-only system that can be copied and distributed at no charge. Some companies sell the run-only systems on an individual basis. Still others sell the rights to producing an unlimited number of run-only systems for a one-time price. Licensing issues can be critical to the cost-effective distribution of the application.

4.1.8 Hardware Requirements

Another key factor in expert system shell selection is the hardware requirements. A shell will not be effective if it cannot be used on the equipment that is available to the developer and the users. It is important to identify the hardware specifications of the shell early in the decision process, including factors such as:

- Memory requirements
- Disk requirements
- Graphics compatibility
- Operating system.

4.1.9 Maintenance

The maintainability of the shell can be another important criteria in shell selection. In order for an expert system to be effective and successful, the developers must be able to maintain it easily, including revising it in response to shell upgrades. Positive technical support on the part of the shell manufacturer is a plus in this aspect of expert system development.

4.2 Consulting Users on User Interface Issues

A vital, but often neglected, facet of expert system development is the user interface. Although there may not be much flexibility in this area when an expert system shell is used, there are usually some options available to the developer. Because the users will be required to interact directly with the expert system on a regular basis, it is important that they are involved in determining the means of that interaction. The developer should talk to the users to determine the most important criteria for the user interface, using section P.2 of the **Prototype Phase Worksheet** (Fig. 4.2) as a guideline. After selecting the features that are important to the users, these items should be ranked in order of importance, for use as reference during the development of the interface. Other important issues to consider when discussing the user interface include:

- What are users familiar with and are currently using?
- How much time is available for the use of the expert system?
- How will it be used (e.g., emergency, quick turn-around, planning, analysis, etc.)?
- Which features are most important to them (e.g., maps and graphics or descriptive text)?
- How will the users be presented with questions and what type of data entry will be involved?
- What level of training and computer experience does the typical user have?
- What are the response requirements - are justifications and rationale needed for a recommendation?
- How critical are the expert system's answers to the users?
- Is an audit trail needed that would require a file to be kept of each session, including the users name, date, and time?

4.3 Knowledge Acquisition Sessions

Knowledge acquisition is the most complex, time-consuming, and unstructured task in developing an expert system. Knowledge acquisition is not a linear process, but more of an iterative process, as shown in Fig. 4.4. After knowledge is elicited from the expert during the interviews, the developer will document and test the knowledge, then convert it into the selected knowledge representation. At each of these stages, gaps and inconsistencies in the knowledge will appear. The developer will then return to the expert to clarify the knowledge and elicit further information. A key to successful knowledge acquisition is proper attention to preparation and

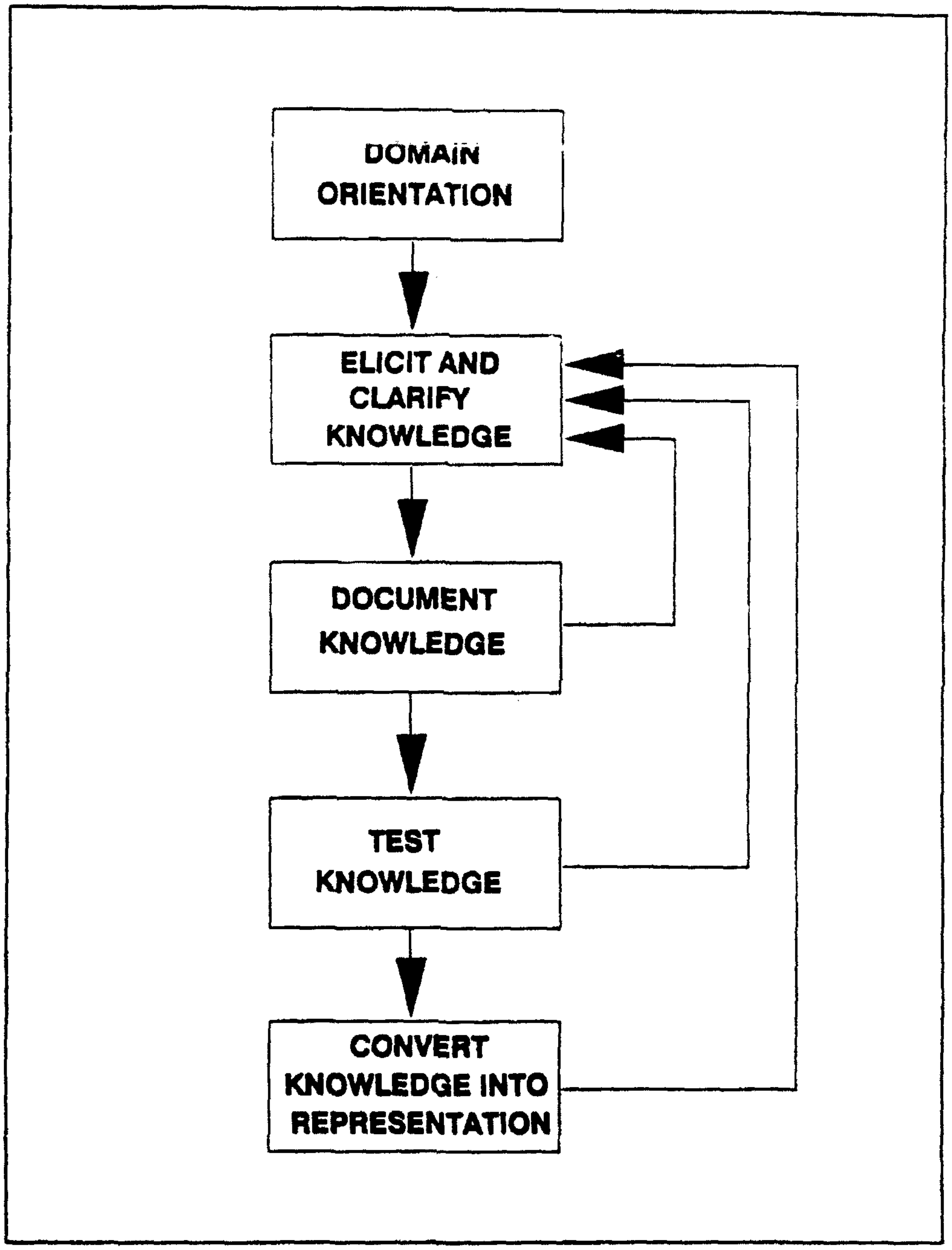

Fig. 4.4 The Knowledge Acquisition Process

detail. This will not only expedite the knowledge acquisition process, but also facilitate both testing and maintenance of the final system.

Decisions regarding methods of eliciting and documenting the knowledge were made during the **Definition Phase**. During the **Prototype Development Phase**, each of these aspects of knowledge acquisition, along with testing the completeness and consistency of the knowledge and converting it into the selected knowledge representation, will be implemented by the knowledge engineer.

4.3.1 Domain and Expert Orientation

In cases where the developer and the expert are not the same individual, the developer should have an overview level of familiarity with the domain before beginning the knowledge acquisition sessions with the expert. This domain orientation can be obtained through reading books and trade journals, observing the expert in action, and talking to people who are familiar with the domain. Complete section P.3 in the **Prototype Phase Worksheet** (Fig. 4.5) as a record of domain orientation, using form P.3.1 **Domain Orientation** to record any important information that is collected.

Following domain orientation, the knowledge engineer should get to know the expert, review knowledge gathered to date, gather the necessary tools and aids, set up a schedule for the knowledge acquisition sessions, and plan the first session. Important tools and aids to be used include interview guides (Fig. 4.7), process flowcharts, and blank tables or graphs. The session planning should involve scheduling, topics to be covered, and a preliminary sequence.

The knowledge engineer should also prepare the expert by explaining the project in some detail, describing what is expected, and motivating the expert. It is important that the expert feels at ease with the process and the role that he or she is playing. The knowledge engineer can aid in this by explaining the importance of the expert in the process, and emphasizing that the purpose of the project is not to replace the expert, but to make his or her expertise more widely available.

4.3.2 Eliciting Knowledge

Once the knowledge acquisition process has begun, it is important to adhere to the initial plan - adapting it where necessary - and to be thorough. For each knowledge acquisition session, a **Knowledge Acquisition Planner** should be completed (Fig. 4.6). This covers the tasks involved in eliciting, recording, compiling, documenting, and analyzing knowledge and preparing for the next session. Mistakes to avoid during knowledge acquisition (adapted from Hart, 1986) include:

- Failing to ask enough questions
- Not adhering to objectives
- Being too general
- Forgetting answers
- Ignoring suggestions
- Making false assumptions
- Misunderstanding jargon.

PROTOTYPE PHASE WORKSHEET (cont.)

P.3 DOMAIN ORIENTATION

- SOURCE OF BACKGROUND DATA IDENTIFIED ✓
- SOURCES USED: Technical reports, simulation documents, informal conversations with experts
- EXPERT(S) OBSERVED AT WORK ✓
- GENERAL CONVERSATIONS WITH EXPERT(S) ✓
- P 3.1 FINDINGS RECORDED ✓

P.4 KNOWLEDGE ACQUISITION SESSIONS

- SCHEDULE DEVELOPED ✓
- EXPERT BRIEFED ON PROJECT ✓
- KNOWLEDGE ACQUISITION PLANNER COMPLETED FOR EACH SESSION ✓

P.5 TEST CASE SELECTION

- TEST CASES IDENTIFIED
  - AVERAGE/FREQUENT CASES ✓
  - EXTREME/SIGNIFICANT CASES ✓
- TEST CASES SET ASIDE FOR LATER TESTING ✓

Prior to the knowledge acquisition sessions, the developer, who was not familiar with the network design process, completed the *domain orientation*. This involved reading a number of documents and technical reports on the subject, and studying a number of completed designs. The developer also observed several experts and informally discussed the design process with them. The expert was briefed on the project, and a schedule of *knowledge acquisition* sessions was set up. For each of the sessions, the developer completed a **Knowledge Acquisition Planner** and an **Interview Guide**. Throughout the knowledge acquisition process, the developer and expert also selected a number of cases to set aside for later use in the **Test and Evaluation Phase**.

Fig. 4.5 Sample Prototype Phase Worksheet (cont.)

KNOWLEDGE ACQUISITION PLANNER
(COMPLETE FOR EACH SESSION)

DATE: 7/2/90 **INTERVIEW NO.:** 3

INTERVIEW GUIDE DEVELOPED	✔
INTERVIEW PERFORMED	✔
KNOWLEDGE RECORDED	✔
KNOWLEDGE COMPILED	✔
MATRICES, DECISION TREES DEVELOPED	✔
KNOWLEDGE ACQUIRED TO DATE ANALYZED	✔
PROBLEMS IN KNOWLEDGE IDENTIFIED	✔

PROBLEMS IDENTIFIED: Missing specific information on light protection. Cannot yet distinguish between when to use moderate or heavy protection. New heavy protection information contradicts original information from first interview.

ISSUES FOR NEXT INTERVIEW: Focus on gathering new light protection information. Clarify heavy protection and specifically contrast it with moderate protection.

DATE AND TIME OF NEXT INTERVIEW: 7/5/90 10 a.m.

Fig. 4.6 Sample Knowledge Acquisition Planner

As the knowledge is being elicited, it is important that it be recorded in an orderly and thorough manner. The knowledge engineer must take careful notes during each interview with the expert, using the **Interview Guide** (Fig. 4.7) as a reference. Although this method can slow down the interview and limit the free flow of information, it is a necessary part of the knowledge acquisition process.

During the knowledge acquisition process, it is helpful to discuss numerous example cases. Many of these cases will be incorporated into the logic of the expert system. At this time, the knowledge engineer should select a number of these cases to set aside for use during the full system testing in the **Test and Evaluation Phase**. Although this may result in a reduction in knowledge, it will aid in the testing of the prototype expert system by providing cases that were not used during the development process, and allowing for effective testing of the expert system's response to new data. Testing scenarios that include average cases, plus extreme cases on the high and low ends to test the boundaries of the expert system are excellent methods to test for outliers and unpredictable results. Testing should also include most frequent cases, most significant cases, and a worst case scenario, which results in an extreme negative impact. Upon the selection of suitable test cases, these items should be checked off on the **Prototype Phase Worksheet**, section P.5 (Fig. 4.5).

It is important to remember that knowledge acquisition is an incremental process. After each knowledge acquisition session, the knowledge engineer will document the knowledge in the chosen form. This will provide a solid basis for the next knowledge acquisition session and will also help the knowledge engineer to analyze and test the knowledge to ensure that the full depth and breadth of the domain is explored. The knowledge engineer should test for missing data, inconsistencies, and loops in the knowledge. These problems should be identified as early as possible, and worked out with the expert to avoid problems later in the development process.

When all of the scheduled knowledge acquisition sessions have been completed and the developer has enough information to provide a basis for the expert system, the knowledge acquisition process has been completed. Item P.4 on the **Prototype Development Phase Checklist** (Fig. 4.5) should be checked off and the date of completion indicated.

4.4 Compiling the Knowledge

Following each knowledge acquisition session, it is important to compile and document the new knowledge, as indicated in the **Knowledge Acquisition Planner** (Fig. 4.6). Forms of documenting the knowledge vary, as discussed in the **Definition Phase**, but should be easy to use and conducive to manipulation and testing. Examples of tables and decision trees at varying stages of knowledge acquisition are shown in Figs. 4.8 - 4.10. Graphical representations such as these should be prepared following each knowledge acquisition session to assist the developer in identifying gaps or discrepancies in the knowledge.

New knowledge should be divided into topics and compared to existing knowledge. After removing redundant and irrelevant knowledge, the remainder is then tested for logical consistency

INTERVIEW GUIDE

DATE: 7/5/90 **INTERVIEW NO.:** 4

EXPERT: Bob Smith

KNOWLEDGE ENGINEER: George Martin

TOPICS TO BE COVERED: Physical Protection of Nodes

ISSUES TO BE RESOLVED: What does light protection consist of? How does heavy protection differ from moderate protection? Are buildings actually buried for heavy protection or not?

NEW KNOWLEDGE: (1) Light protection includes controlled access and auxiliary power.
(2) Heavy protection has all of the safe-guards of moderate protection and adds that all operations are buried.
(3) Nodes X, Y, and Z are all examples of heavy protection.
(4) Reconstitution issues should be the topic of next session.

Fig. 4.7 Sample Interview Guide

NODE IS OVERRUN	NODE IS COLOCATED W/ CRITICAL USERS	AT LEAST ONE CRITICAL USER IS SURVIVING	NODE IS A MEMBER OF 2 OR MORE NETWORKS	LEVEL OF HARDENING	RECOMMENDED LEVEL OF PROTECTION
				NONE	MINIMAL
NO	YES	YES		MINIMAL	LIGHT
NO	YES	YES	YES	LIGHT	MODERATE
NO	YES	YES	YES	MODERATE	HEAVY

Fig. 4.8 Sample Decision Table After First Knowledge Acquisition Session

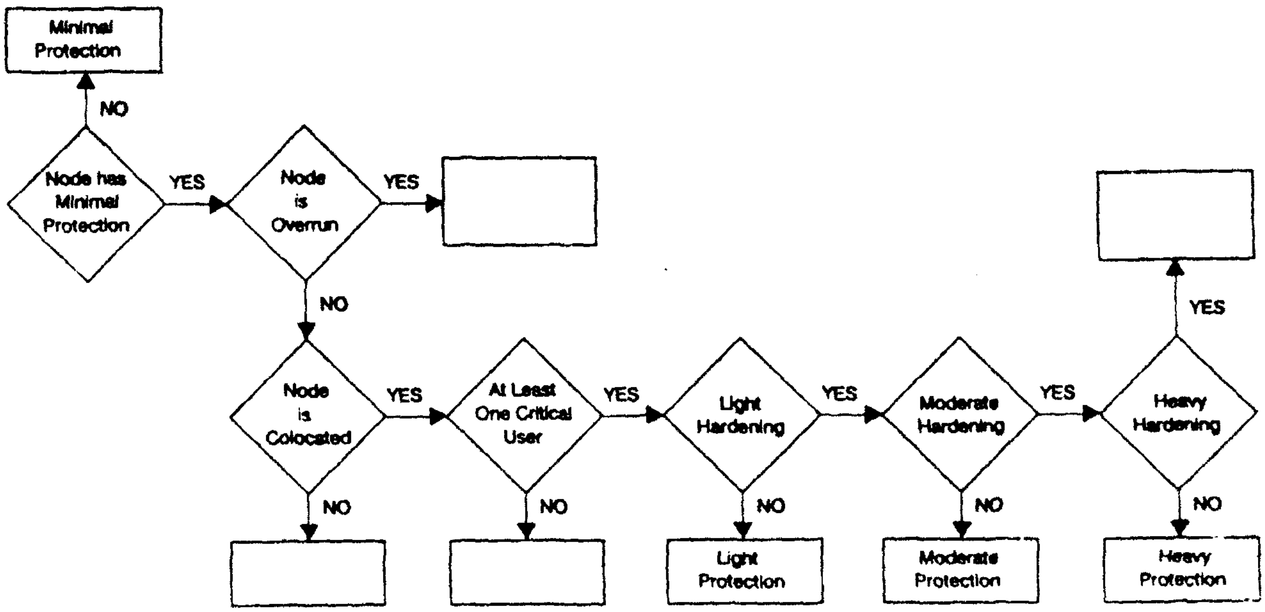

Fig. 4.9 Sample Decision Tree After First Knowledge Acquisition Session

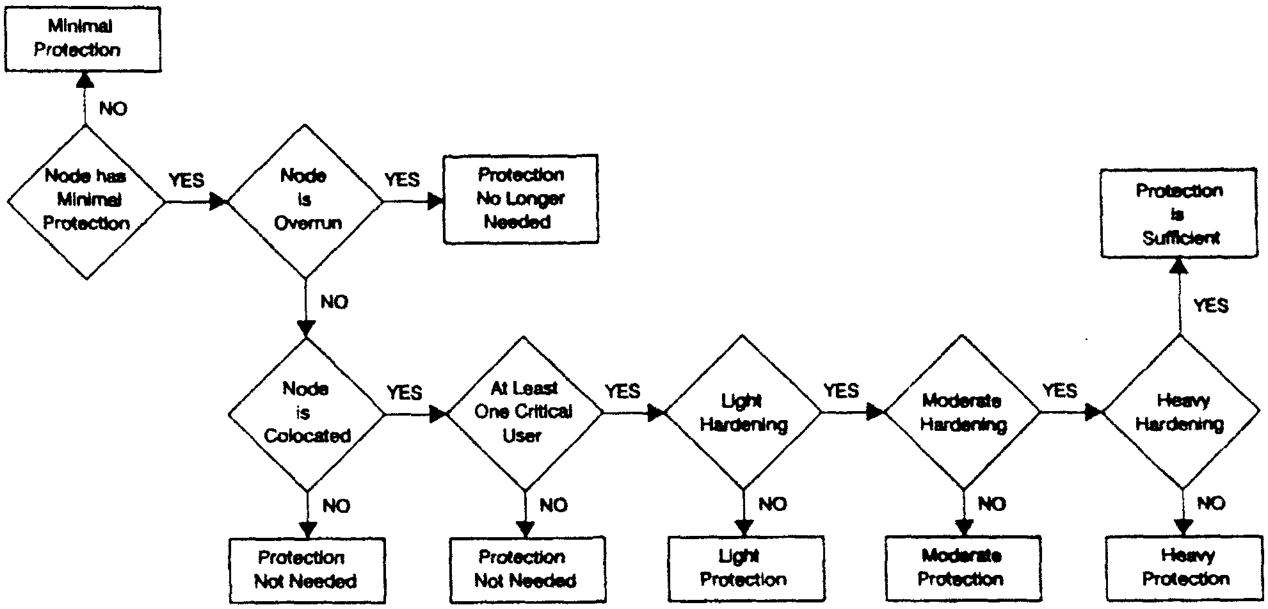

Fig. 4.10 Sample Decision Tree After 2nd Knowledge Acquisition Session

with existing knowledge. The purpose is to identify and classify knowledge into one of the following categories:

- New and unique information
- A generalization or specialization of existing information
- An assumption
- Contradictory to previous information.

As new knowledge is compiled, it is a continuing process to identify and correct discrepancies and gather missing knowledge. Missing knowledge can be classified in terms of breadth and depth. Breadth refers to the different topics within the problem, while depth concerns the detail in which each topic is covered. Missing breadth knowledge often arises from topics newly discovered or not yet considered. Missing depth knowledge, on the other hand, often results from lack of detail or omitted steps within a process.

Once missing knowledge has been identified, two steps must occur. The first is to determine whether or not it is important and lies within the scope of the current system. If so, the second step is to allocate a portion of a future knowledge acquisition session to gathering the missing information.

4.5 Converting the Knowledge into the Selected Knowledge Representation

Once a sufficient body of knowledge has been compiled and analyzed, the final step in the knowledge acquisition process is to convert the knowledge into the appropriate knowledge representation, whether it be in the form of rules or examples. The conversion of knowledge is generally the responsibility of the programmer, who takes knowledge from the decision trees or matrices, and puts it into the format required by the shell that is being used. During the conversion process, the developer may find additional problems with the data that require further clarification by the expert.

Because of the poorly structured nature of knowledge engineering, it is normal for the knowledge acquisition and conversion process to continue well into the expert system development process. In addition to adding new topics and functionality, further knowledge engineering refines the existing system to more closely emulate the behavior of the expert.

4.6 Performing Incremental Testing and Review of the System

Throughout the development process, incremental testing of the expert system must be performed. Just as a conventional programmer would test ten individual subsections of one hundred lines of code before combining them into a one thousand line program, the expert system developer tests the system module by module. Errors can be tracked and corrected more readily using an incremental testing process. Frequent backups of the expert system knowledge base should be made and versions labeled and archived so the developer can retrieve a past version that may perform better than after a change was made.

Testing the knowledge base for structural and content items, as well as testing the performance and usability of the expert system (see Appendix A) during this phase will improve the expert system, and also prepare it for testing during the **Test and Evaluation Phase**. Tests for completeness, accuracy, and consistency are very important in this step. Frequently, careful documenting of the knowledge identifies areas of omission and conflicting data. Several iterations may be required to develop a complete base of knowledge for the application. The knowledge engineer should analyze the knowledge after each knowledge acquisition session, looking for problems, then present any issues that arise to the expert for resolution. The expert is responsible for reviewing the logic prepared by the knowledge engineer to ensure that the knowledge has been interpreted correctly.

At this point in the development process, it is also important to re-evaluate the **Requirements Statement** (Fig. 3.2) and the design documents to ensure that system development stays on track, as indicated in the **Prototype Phase Worksheet**, sections P.6 and P.7 (Fig. 4.11). Any discrepancies should be remedied, either by revising the expert system if items have been overlooked and can reasonably be included, or by revising the documents if they have become outdated or incorrect.

PROTOTYPE PHASE WORKSHEET (cont.)

P.6 REQUIREMENTS STATEMENT REVIEWED

- REQUIREMENTS REMAIN FEASIBLE ☑
- PROJECT STILL CONFORMS TO REQUIREMENTS ☑
- CONFLICTS IDENTIFIED ☑

  Users would like more control over running the expert system, but will settle for bimonthly upgrades.
- DOCUMENT MODIFIED ACCORDINGLY ☑

P.7 DESIGN DOCUMENTS REVIEWED

- DESIGN REMAINS ACCURATE ☑
- PROJECT STILL CONFORMS TO DESIGN ☑
- CONFLICTS IDENTIFIED ☑

  None
- DOCUMENTS MODIFIED ACCORDINGLY ☑

P.8 DOCUMENTATION PREPARED

USER'S MANUAL

- BACKGROUND/OVERVIEW ☑
- INTENDED USERS ☑
- ACCESSING SYSTEM ☑
- INTERFACE FORMAT ☑
- USING THE SYSTEM ☑
- FILES INVOLVED ☑
- SAMPLE SESSIONS ☑

DEVELOPER'S NOTEBOOK

- REQUIREMENTS STATEMENT ☑
- SYSTEM DESIGN ☑
- DEVELOPMENT WORKSHEETS ☑
- KNOWLEDGE ACQUISITION DATA ☑
- MATRICES, GRAPHICS ☑
- FILE PROCESSING ☑
- MAINTENANCE ISSUES ☑
- PROJECT MANAGEMENT AIDS ☑

During the **Prototype Development Phase**, the **Requirements Statement** and design documents were reviewed. They were found to be acceptable, with the project remaining on course and conforming to the requirements. The documentation was also prepared at this time by the developer.

Fig. 4.11 Sample Prototype Phase Worksheet (cont.)

4.7 Preparing Documentation for the Prototype

There are typically two types of documentation that should accompany a small expert system application:

- User's Manual
- Developer's Notebook.

The **User's Manual** provides guidance on how to operate the expert system, and accompanies all copies of the expert system that are distributed to users. The **Developer's Notebook** is designed to keep a record of the development process, and is used by the individual assigned to maintain the system, and also by other expert system developers who can use it as an example for future development. Use section P.8 in the **Prototype Phase Worksheet** (Fig. 4.11) as a checklist for the preparation of the documentation. Upon completion of the documentation, check off item P.8 on the **Prototype Development Phase Checklist** (Fig. 4.1) and indicate the date of completion.

4.7.1 User's Manual

The **User's Manual** describes the normal operation of the expert system from a user's perspective. It contains all of the information needed to run the application, including:

- Navigating the system
- Help functions
- Files used
- Function keys
- Command words
- User interface
- Interacting with the system
- Processing that occurs
- Outputs produced
- Saving files.

It should also include general information about the expert system, such as:

- Purpose and objectives of the system
- Assumptions made during development
- Target users
- Names of development team members
- Date of development
- Where to go for help
- System outputs.

The **User's Manual** should also include a sample session, showing all of the screens that will be encountered and the responses requested while running the system. Any information needed on external processing and file management should also be included in the **User's Manual**.

4.7.2 Developer's Notebook

The **Developer's Notebook** contains key facets of the development of the expert system. It includes such things as:

- Problem assessment, including paths considered but not taken
- **Requirements Statement**
- Expert system design
- Decision trees and other diagrams
- Completed checklists and worksheets from this guidebook
- Interview guides and other forms
- Management aids - Gantt charts, PERT charts, etc.
- Documented code
- Test cases
- Verification and validation procedures
- Instructions on system maintenance
- Names of development team members
- Comments made throughout development and testing
- Distribution issues and assumptions.

For the small expert systems that are being defined, some of these items may be only one or two paragraphs. The **Developer's Notebook** is a useful tool for those responsible for the maintenance of the expert system. It also serves as an effective guide for others interested in developing expert systems, because it documents the entire development process.

4.8 Summary

Upon completion of the **Prototype Development Phase**, many tasks have been performed in the development of a prototype expert system. Each of these tasks, from the selection of a development tool, to the preparation of the system documentation, has been checked off on the **Prototype Development Phase Checklist**. When the checklist is completed, the milestone of the production of a prototype expert system is achieved.

5.0 The Test and Evaluation Phase

After the prototype version of the expert system is completed, it must be carefully tested and evaluated. Although the expert system has been tested incrementally throughout the **Definition Phase** and the **Prototype System Development Phase**, it must now be fully tested before it is ready for distribution. The expert and developer will be responsible for verification of the content and logic of the system, and the users will be responsible for validating the system as a whole. VERIFICATION ensures that the expert system has been developed correctly and does not contain technical errors. VALIDATION ensures that the expert system is useful and satisfies the users' needs.

The **Test and Evaluation Phase Checklist** (Fig. 5.1) serves as a guide for the testing and evaluation of the expert system. The testing issues addressed during this phase relate to a number of areas in the **MAU Framework** (Appendix A), including the knowledge base, service issues, performance, and usability of the expert system.

5.1 Verification

The expert system will be verified by the expert to ensure that the rules are appropriate, correct questions are asked and the correct conclusions are reached. Because it may be difficult for the expert to follow the format of the knowledge base, it can be useful for the expert to perform a desk review of the knowledge base, with the knowledge engineer available to explain the rules. The expert can also verify the expert system using the test cases that were set aside for this purpose during the knowledge acquisition process. Additional test cases can also be developed by the expert at this time for testing purposes. While testing the expert system, the expert should be looking for missing data, incorrect conclusions, correct conclusions that were improperly derived, and any other discrepancies. The expert should keep a log of the testing on the **Expert Evaluation Form** (Fig. 5.2), recording any problems or comments that arise.

While the expert is testing the knowledge, the developer is responsible for verifying the logic and inferencing mechanisms of the expert system. The developer checks the system for loops in the logic, isolated rules, and cases where a conclusion is never reached. The developer can work directly with the knowledge base, and also run the system to test a variety of scenarios. The use of an expert system shell can reduce the amount of testing required by the knowledge engineer due to the existence of an established inference engine, but this should not simply be assumed to be correct.

The use of an expert system shell gives the developer access to a number of tools that assist in the testing process. Shells often contain a number of useful functions that allow the developer to observe the rules as they are being fired, query the system as to why a particular conclusion was reached, and perform many other useful diagnostic tasks. Such features allow the knowledge engineer to easily trace the logic and verify that it is operating correctly, or identify the locations where revisions are warranted. Throughout the verification process, the knowledge engineer should record all findings on the **Developer Evaluation Form** (Fig. 5.3) for use in the **Final Development Phase**.

TEST AND EVALUATION PHASE CHECKLIST

VERIFICATION BY EXPERT		
PERSON RESPONSIBLE		Bob Smith
		DATE
TESTING COMPLETED	✔	8/6/90
FINDINGS RECORDED	✔	8/6/90
E.1 EXPERT EVALUATION FORM COMPLETED	✔	8/6/90
VERIFICATION BY DEVELOPER		
PERSON RESPONSIBLE		George Martin
TESTING COMPLETED	✔	8/8/90
FINDINGS RECORDED	✔	8/8/90
E.2 DEVELOPER EVALUATION FORM COMPLETED	✔	8/8/90
VALIDATION OF OVERALL SYSTEM BY USERS		
TEST DISTRIBUTED	✔	8/6/90
TESTING COMPLETED	✔	8/10/90
FINDINGS RECORDED	✔	8/10/90
E.3 USER'S EVALUATION FORMS COMPLETED	✔	8/10/90
EVALUATION OF USER'S MANUAL		
E.4 USER'S MANUAL EVALUATION FORMS COMPLETED	✔	8/10/90
EVALUATION OF DEVELOPER'S NOTEBOOK		
E.5 DEVELOPER'S NOTEBOOK EVALUATION FORMS COMPLETED	✔	8/9/90
REQUIREMENTS AND DESIGN REVIEWED		
REQUIREMENTS STATEMENT REVIEWED	✔	8/9/90
DESIGN DOCUMENTS REVIEWED	✔	8/9/90
MILESTONE		
TESTING AND EVALUATION COMPLETED	✔	8/10/90

During the **Test and Evaluation Phase**, **ESTEL** was carefully verified and validated by the expert, developer, and users, and their comments were recorded. The documentation was also evaluated, and the requirements and design were reviewed.

Fig. 5.1 Sample Test and Evaluation Phase Checklist

E.1 EXPERT EVALUATION FORM

EXPERT: Bob Smith **DATE:** 8/6/90

TEST CASES APPLIED TO EXPERT SYSTEM

		NUMBER
AVERAGE/FREQUENT CASES	☑	20
EXTREME/SIGNIFICANT CASES	☑	10

PROBLEMS IDENTIFIED

KNOWLEDGE Microwave LOS links have 384 channels with multiples, not 672 channels

LOGIC Should determine all physical protection issues before moving to electronic. Asked 2 irrelevant questions.

ORDER OF QUESTIONING OK

OVERALL IMPRESSION LOW → HIGH

__ __ __ ✓ __

COMMENTS Would like to try several more extreme cases when new data arrives next month.

Fig. 5.2 Sample Expert Evaluation Form

E.2 DEVELOPER EVALUATION FORM

NAME: George Martin **DATE:** 8/8/90

TEST CASES APPLIED TO EXPERT SYSTEM

		NUMBER
AVERAGE/FREQUENT CASES	☐	30
EXTREME/SIGNIFICANT CASES	☐	15

PROBLEMS IDENTIFIED

LOGIC 2 questions are never asked in physical protection - a rule may be isolated.

INFERENCING OK

INTERFACE Justification interface is slow, cumbersome.

COMMENTS Need to redesign justification rules to operate more efficiently.

Fig. 5.3 Sample Developer Evaluation Form

5.2 Validation

While it is important that the expert system is tested internally by the expert and the developer, it is also critical that it be tested from a user's perspective. It is recommended that a small group of users participate in the testing process. This group will be responsible for evaluating the overall usefulness of the system, including items such as:

- User interface - screens, graphics, menus, etc.
- System response time
- User-friendliness
- Appropriateness of subject level
- File management issues.

The developers should send the users guidelines or forms to use in testing the expert system. This will provide the users with instructions on testing, and aid them in recording their evaluations of the system. Each user will record all comments and suggestions about the expert system on the **User Evaluation Form** (Fig. 5.4) and return it at the conclusion of the testing period.

5.3 Evaluation of the Documentation

An additional task to be performed during the **Test and Evaluation Phase** is the evaluation of the documentation. Although it is not as apparent, the documentation is a vital part of the expert system. If the users do not understand how to operate the system, it cannot be used effectively. The **User's Manual** should provide complete instructions on the operation of the system and provide users with assistance when needed. The **Developer's Notebook** is critical to the maintenance of the expert system.

The users are responsible for evaluating the **User's Manual**. They should focus on features such as:

- Readability
- Accuracy
- Clarity
- Completeness
- Ease of understanding.

Comments and suggestions on the documentation will be recorded in the **User's Manual Evaluation Form** (Fig. 5.5) for use in the **Final Development Phase.**

E.3 USER EVALUATION FORM

NAME: Susan Wallace **DATE:** 8/7/90

COMMENTS

INTERFACE Functions all operate properly. Some text is incorrect and needs to be reworded. Too slow!

RESPONSE TIME Fair to poor.

USER-FRIENDLINESS Easy for experts to use, but needs more help functions for novices.

APPROPRIATENESS OF SUBJECT LEVEL Good

FILE MANAGEMENT ISSUES N/A

GENERAL Overall performance is good. Should help speed up design process.

OVERALL IMPRESSION LOW → HIGH

__ __ __ ✓ __

Fig. 5.4 Sample User Evaluation Form

E.4 USER'S MANUAL EVALUATION FORM

NAME: Fred Arthur **DATE:** 8/9/90

OVERALL DOCUMENT

	YES	NO
READABLE	✓	
ACCURATE	✓	
CLEAR		✓
COMPLETE	✓	
EASILY UNDERSTANDABLE		✓

COMMENTS Needs more explanation on data used by ESTEL and how to operate justification.

SAMPLE SESSIONS

	YES	NO
UNDERSTANDABLE	✓	
ACCU'RATE	✓	
CLEAR	✓	
COMPLETE	✓	
DESCRIPTIVE	✓	

COMMENTS

OVERALL IMPRESSION

LOW → HIGH

		✓		

Fig. 5.5 Sample User's Manual Evaluation Form

The developers and manager are responsible for evaluating the **Developer's Notebook** and recording their comments on the **Developer's Notebook Evaluation Form** (Fig. 5.6). This document should be evaluated for the same features that are listed above for the **User's Manual**. In addition, an evaluation should be made of the supporting information that makes up this document, including:

- **Requirements Statement**
- System Design
- Development worksheets
- Knowledge acquisition data
- File processing and maintenance issues.

5.4 Review of System Requirements and Design

To complete the evaluation, the prototype expert system should be compared to the **Requirements Statement** and the design documents. To avoid "requirements drift," these documents were used in the incremental testing process during the **Prototype Development Phase**. They become even more important at this stage to calibrate the extent to which the prototype system meets the original expectations. System success is evaluated based on the qualitative and quantitative benefits documented during the **Concept Phase** (Fig. 2.1). Benefits can be measured in terms of the savings in time, staff-hours, costs, etc. that are achieved.

5.5 Summary

At the completion of the **Test and Evaluation Phase**, all of the items on the **Test and Evaluation Phase Checklist** have been completed, using the **MAU Framework** (Appendix A) as a guide. The completion of the checklist indicates that the testing and evaluation of the expert system and supporting documentation is finished, and the requirements for the milestone have been met.

E.5 DEVELOPER'S NOTEBOOK EVALUATION FORM

NAME: Mary Williams **DATE:** 8/8/90

OVERALL DOCUMENT

	YES	NO
READABLE	✓	
ACCURATE	✓	
CLEAR	✓	
COMPLETE	✓	
EASILY UNDERSTANDABLE		✓

COMMENTS Add more text on what the components mean, how the development process progressed.

SUPPORTING DOCUMENTS INCLUDED

	YES	NO
REQUIREMENTS STATEMENT	✓	
SYSTEM DESIGN DOCUMENTS	✓	
DEVELOPMENT WORKSHEETS	✓	
KNOWLEDGE ACQUISITION MATERIALS		✓
FILE PROCESSING AND MAINTENANCE ISSUES		✓
PROJECT MANGEMENT AIDS	✓	

COMMENTS Seems to be missing some diagrams from knowledge acquisition. Does not address file processing.

OVERALL IMPRESSION LOW → HIGH

__ __ __ ✓ __

Fig. 5.6 Sample Developer's Notebook Evaluation Form

6.0 The Final Development Phase

At the completion of the **Test and Evaluation Phase**, the developers receive comments from each group involved in the testing process - the expert, the developers, and the users. During the **Final Development Phase**, these comments are evaluated and incorporated into the final version of the expert system, using the **Final Development Phase Checklist** (Fig. 6.1) as a guideline.

6.1 Evaluating User Comments

The first step in this phase is to evaluate the user comments, and decide which suggestions will be incorporated into the expert system. Because the user interface is somewhat subjective, it is quite possible that one user might like a feature that another user finds objectionable. If there are conflicts such as this, it is best to go with the majority opinion or with the project sponsor. If there are any major issues, it may be useful to contact the users involved in the testing to clarify the matter.

Some changes that are suggested may be impossible to incorporate, given the limitations of the shell used to develop the expert system. Other changes may simply be infeasible, given the time and resources required to implement them.

To effectively incorporate the user comments, the developers should first prioritize the suggested changes, with functionality and performance issues being ranked highest, and appearance and interface issues being ranked lower on the list. The developers can then estimate the resources required to make the suggested revisions. This information will be presented to the project manager, who will determine which of the comments will be incorporated into the final system, using Fig. 6.2 as a guide.

6.2 Incorporating the Changes

When the user comments have been evaluated and appropriate changes have been selected for inclusion, these changes, as well as those noted by the expert and developer during their testing, will be incorporated into the system. It is critical that any problems in the rules or logic be corrected in order for the system to run correctly. But it is also important to incorporate as many of the user comments as possible, to ensure their satisfaction with the system.

After changes are made to the expert system, the developer will test it again to ensure that all of the previously identified problems have been eliminated and no new ones have been created. This testing period will be shorter, focusing on the portions of the expert system that were changed, and will not include the users.

During the **Final Development Phase**, any suggested changes to the **User's Manual** are also be incorporated into the final version, and any items missing from the **Developer's Notebook** incorporated into this document. At the completion of this phase, the expert system should be adequately tested and evaluated, and the system ready to be prepared for distribution.

FINAL DEVELOPMENT PHASE CHECKLIST

EVALUATION RESULTS COLLECTED

		DATE COMPLETED
END-USERS	☑	8/10/90
EXPERT	☑	8/10/90
DEVELOPER(S)	☑	8/10/90
OTHERS	☐	

END-USER COMMENT EVALUATION

COMMENTS COMPILED	☑	8/13/90
COMMENTS ANALYZED	☑	8/14/90
RELEVANT & FEASIBLE COMMENTS SELECTED	☑	8/14/90

CHANGES INCORPORATED INTO EXPERT SYSTEM

END-USERS	☑	8/17/90
EXPERT	☑	8/16/90.
DEVELOPER(S)	☑	8/15/90
OTHERS	☐	

EXPERT SYSTEM RETESTED

RULES VERIFIED	☑	8/22/90
LOGIC VALIDATED	☑	8/22/90
INTERFACE REVIEWED	☑	8/22/90

CHANGES INCORPORATED INTO DOCUMENTATION

END-USERS	☑	8/13/90
KNOWLEDGE ENGINEER	☑	8/15/90
OTHERS	☑	8/15/90

MILESTONE

FINAL EXPERT SYSTEM PRODUCED	☑	8/22/90

During the **Final Development Phase**, the results of the evaluations of ESTEL were collected and evaluated. Relevant and appropriate modifications and enhancements were incorporated into the system. The developer then retested the system to ensure that the changes were properly implemented. Comments on the documentation were also compiled and incorporated into the documents.

Fig. 6.1 Sample Final Development Phase Checklist

THIS

PAGE

IS

MISSING

IN

ORIGINAL

DOCUMENT

6-3

THIS
PAGE
IS
MISSING
IN
ORIGINAL
DOCUMENT

7-1 & 7-2

- **Policy** - the policies that a system is based on may change over time.

A periodic review of the system should be undertaken every three months to one year, depending on how dynamic the domain is. If changes are extensive, a comprehensive system rewrite may be required. Information on system reviews and upgrades should be indicated on the **Post-Development Phase Checklist** (Fig. 7.1).

An issue related to system maintenance and upgrades is version control. Whenever the expert system is changed, a new official version will be distributed to all users, along with updated documentation. At this time, all previous versions are returned to the developers or destroyed, to ensure that all users are operating the same version of the expert system. This process can be facilitated by assigning a registration number to each copy of the expert system that is distributed and recording the number assigned to each user (see section 7.4).

7.2 Technical Support

The individual responsible for developing the expert system will also provide technical support to the users upon distribution of the system. If the users are in the same location as the developers, no special plans may be needed. However, if the users are widespread geographically, it may be necessary to set up a technical support hotline. The technical support person will be responsible for several areas, including:

- Providing information on how to use the system
- Resolving hardware and interface problems
- Recording bugs and problems reported by the users.

7.3 Training

Training is also an important component of a successful expert system implementation. The goal is that the users be knowledgeable about the expert system and software in order to use it effectively. A number of user training approaches can be taken, depending on several factors, such as the:

- Number of users
- Complexity of the system
- Level of required precision
- Users' level of expertise in the subject area
- Volatility of the knowledge base
- Users' level of computer experience.

For simple expert systems, training may be unnecessary beyond the user's manual, accompanied by a tutorial or demo disk showing a sample session. But in many cases, it is beneficial to provide more in-depth training to ensure that the users are comfortable and familiar with the expert system. Training can be provided on an individual basis or for groups. Once a training strategy has been selected, the trainer must prepare for it accordingly, developing a

training manual, tutorial disk, and training presentation, based on the users' knowledge of the subject area and familiarity with computers.

There are a number of important issues to cover in the training sessions. First, it is useful to explain the purpose of the expert system and its role within the current process. The users will want to know how the addition of the expert system will affect their everyday duties and responsibilities. The trainer should also define the intended users and the level of computer and subject area expertise that is assumed. A major issue that is often left out of the training process is informing the users of the expert system's intended use, and stressing the risks of improper use of the system. Any expiration date of the system should also be mentioned during training.

7.4 Distribution

Depending on the number of users of the expert system, distribution can range from a simple task to a very large operation. In either case, several steps must be taken for it to be implemented smoothly. A database should first be set up to register the users, containing the names, addresses, and phone numbers of all users. Each user is contacted to identify the type of media his or her hardware supports, e.g., high or low density, 5-1/4" or 3-1/2" disks. This data is also recorded in the user database. Preparations are also made for the packages that will be sent out to users, taking into consideration the following issues:

- Ordering disks
- Formatting disks
- Producing labels with registration numbers
- Preparing run-only copies of software, if applicable
- Copying files onto disks
- Preparing letter to accompany expert system
- Producing final documentation
- Preparing packaging for expert system.

While distribution plans are being made, any licensing issues related to the expert system shell will be resolved. Licensing issues related to distributing an expert system were first addressed during the **Prototype Development Phase**, when the expert system development tool was selected. At this time, any applicable run-only software will be ordered or prepared, and any remaining licensing issues must be resolved.

7.5 System Implementation

With sufficient preparation, the implementation of the **Post-Development Phase** steps will proceed smoothly. Distribution includes preparing the expert system packages, sending the packages to the users, and recording registration numbers. Training sessions will be provided to the users within the first few weeks of implementation. It may be beneficial to allow the users a short time to get acquainted with the expert system on their own before training begins, but the users will lose interest in the system if too much time passes before training is provided. The individual responsible for technical support should also be readily available and responsive during the early weeks of expert system use, as users will encounter frequent problems with system installation and operation during this time.

After the initial enthusiasm that accompanies the distribution of the expert system passes, it is important that the development team continues to support the project, by maintaining an interest in improving its performance and responding to the users. The success of an expert system does not depend exclusively on the distribution and acceptance of a working system. The desired benefits of the expert system that were identified in the **Concept Phase** can only be achieved through the continued support of the developers as well as the users.

7.6 Summary

During the **Post-Development Phase**, a number of tasks are completed, including planning the technical support and maintenance strategies, training the users, and distributing the final system. When the **Post-Development Phase Checklist** is completed, with the exception of information on system upgrades, the expert system has been implemented and the requirements for the milestone have been met. However, this is not the end of the development process, as technical support, maintenance, and upgrades of the expert system will continue for as long as the system is in use.

GLOSSARY

A

advisory system - An alternative name for an expert system.

artificial intelligence - The subfield of computer science which includes expert systems, robotics, and neural networks. Artificial intelligence is "concerned with the concepts and methods of symbolic inference by a computer and the symbolic representation of the knowledge to be used in making inferences. A field aimed at pursuing the possibility that a computer can be made to behave in ways that humans recognize as 'intelligent behavior' in each other" (Feigenbaum and McCorduck, 1983).

B

backward chaining - An inferencing mechanism in which the order that the inferences are drawn is based on the conclusion. Backward chaining begins with a solution and attempts to prove or disprove it by searching through the rules or examples in the knowledge base. This is an interactive form of inferencing that focuses on relevant information and poses only necessary questions.

breadth - The scope of a problem in terms of the different topics or aspects involved.

C

chain of reasoning - A means of making decisions in which topics build on one another and depend on each other for data.

Concept Phase - The first phase in expert system development, in which the problem area is selected and analyzed to determine if an expert system would be appropriate for the application.

conflict resolution strategy - A strategy used in the knowledge acquisition process to resolve discrepancies in knowledge that is collected from several experts.

core development team - The central figures in the expert system development process. This team generally consists of one or more persons performing each of the following roles: expert, knowledge engineer, programmer/software engineer, and manager.

D

decision support system - An alternative name used for an expert system that aids in a decision process.

decision table - A form of knowledge representation in which the information is placed in a table or matrix.

decision tree - A form of knowledge representation in which the information follows each decision point down various paths to reach a conclusion.

Definition Phase - The second phase in expert system development, in which the scope and purpose of the system are carefully defined, the development is selected, and an initial system design is produced.

depth - The scope of a problem in terms of the details involved with each topic or aspect.

desk review - A review performed by reading through the source code of the expert system, rather than running the software.

detailed design - A document prepared during the Definition Phase, which includes a detailed description of each component of an expert system.

documentation - The text that accompanies a completed expert system application and provides information on various aspects of the expert system. Typical documentation for a small expert system includes a User's Manual and a Developer's Notebook.

domain - The subject area in which the knowledge for an expert system resides.

domain orientation - A period of time in which the knowledge engineer becomes acquainted with the domain through background reading and preliminary interviews. This process occurs prior to the knowledge acquisition sessions.

E

example-based - A form of knowledge representation in which numerous example situations are developed and used to form the basis of the decision process for an expert system.

expert - A person who is very familiar with and proficient in a subject area and whose knowledge is used as the basis for an expert system.

expert problem-solver - An alternative name for an expert system.

expert system - A computer system that performs at or near the level of a human expert. Expert systems are often used to capture the knowledge of an individual who is leaving an organization, or to capture a complex or tedious decision-making process so that it can be performed by non-experts, thus reducing the need for training.

expert system development tool - An alternative name for an expert system shell.

expert system shell - A software package which contains the basic components needed to develop an expert system, including the: user and developer interfaces, knowledge representation, inferencing mechanism, debugging facilities, and help functions. The developer uses an expert system shell to enter the expertise into the knowledge representation and quickly develop a working prototype system.

F

Final Development Phase - The fifth phase in expert system development, in which comments from those involved in testing the system are evaluated and incorporated into the system, the final system is prepared for distribution, and the final documentation is prepared.

forward chaining - An inferencing mechanism in which a data or event driven approach is used to search through the knowledge base. In forward chaining systems, the user enters a series of data and the expert system deduces its conclusions from the data available.

H

heuristic - A "rule-of-thumb" or piece of knowledge that is acquired through experience in a particular area, and is often not documented.

I

inferencing mechanism - The means by which an expert system uses information to make decisions and reach conclusions. Common inferencing mechanisms are forward and backward chaining, and are included as part of an expert system shell.

K

knowledge acquisition - The process in which the knowledge engineer collects the expertise that forms the basis for an expert system. This is generally accomplished by interviewing or surveying one or more experts, or by studying documents on the subject area.

knowledge base - The basis for an expert system that has been converted into the selected knowledge representation.

knowledge-based system - An alternative name for an expert system.

knowledge engineer - The member of an expert system development team who is often responsible for designing the system, performing knowledge acquisition, compiling the knowledge, and converting it into the appropriate knowledge representation.

knowledge map - A form of documenting the knowledge within an expert system which provides a graphical representation of the decision process.

knowledge representation - The means by which the knowledge within an expert system is represented. The most common forms are rules and examples.

M

multiattribute utility (MAU) analysis - A methodology based in the mathematics of measurement that provides an appropriate procedure for evaluating expert systems.

O

one-page design - A document prepared during the Definition Phase which provides an overview of an expert system. It includes a description of the system components, basic system structure, interfaces and external calls, and the selected knowledge representation and inferencing mechanism.

P

Post-Development Phase - The final phase in expert system development, in which the full system is distributed to users, and training, technical support, and system maintenance are provided.

proponent - An individual who strongly supports an expert system project. This person may be a member of the expert system development team, or may be outside of the development process.

prototype - An initial version of an expert system which is developed to test the expected goals and requirements of the system. The prototype goes through a number of iterations and tests, then is developed into a full expert system.

Prototype Development Phase - The third phase in expert system development, in which knowledge is collected and developed into the knowledge base of the expert system, and a prototype version of the system and documentation are developed.

R

rapid prototyping - The process of quickly developing knowledge, or a portion of the knowledge, into a prototype expert system, in order to test the concept and requirements of the system.

requirements drift - A common occurrence in which the requirements of an expert system change during the development process as need change or new criteria arise.

Requirements Statement - A document prepared during the Definition Phase which defines the overall expert system in terms of scope, purpose, intended users, capabilities, limitations, output, expectations, and maintenance areas.

rule-based - A knowledge representation in which the information and decision process are represented in an IF/THEN/ELSE format. Rules can be very short, containing only single IF and THEN components, or may be quite complex, consisting of multiple statements with AND/OR qualifiers.

run-only software - A version of an expert system shell which contains only a user interface, and does not have access to the source code, knowledge representation, or inferencing mechanisms.

T

technical support - Support on installing, using, and upgrading the system that is provided to the users by an appointed party, often someone involved in the development process.

Test and Evaluation Phase - The fourth phase in expert system development, in which the system is tested, verified, and validated by users, developers, and experts.

test case - Sample situations that are set aside and kept out of the development process, in order to be used during the Test and Evaluation Phase to ensure that the expert system works properly.

U

upgrades - Revised versions of the expert system that are developed due to software improvements, bug corrections, policy changes, etc.

user interface - The portion of the expert system that the user sees and interacts with. This may include data entry, queries, graphics, reports, etc.

V

validation - The process of testing an expert system to ensure that it is appropriate and usable from a user's perspective.

verification - The process of testing an expert system to ensure that it operates properly. The expert is responsible for verifying that the correct conclusion is reached for each situation, and the knowledge engineer is responsible for verifying that the logic and inferencing mechanisms are working properly.

version control - The process of ensuring that all users have the current version of an expert system when upgrades or new versions are distributed.

BIBLIOGRAPHY

Adelman, Leonard and Jacob W. Ulvila. *Testing and Evaluating Expert Systems.* Reston, VA: Decision Science Consortium, Inc., (1990).

Barr, Avron, Paul R. Cohen, and Edward A. Feigenbaum, *The Handbook of Artificial Intelligence, Volume IV*, Addison-Wesley Publishing Co., Reading, MA, (1989).

De Salvo, Daniel A., and Jay Liebowitz, ed., *Managing Artificial Intelligence and Expert Systems*, Yourdon Press, Englewood Cliffs, NJ, (1990).

Feigenbaum, E.A., and McCorduck, P. *The Fifth Generation*, Addison-Wesley Publishing Company, Reading, MA, (1983).

Harmon, Paul and David King, *Expert Systems*, John Wiley & Sons, New York, NY, (1985).

Hart, Anna, *Knowledge Acquisition for Expert Systems*, McGraw-Hill Book Company, New York, NY, (1986).

Hayes-Roth, Frederick, Donald A. Waterman, and Douglas B. Lenat, ed., *Building Expert Systems*, Addison-Wesley Publishing Co., Reading, MA, (1983).

Hertz, David Bendel, *The Expert Executive*, John Wiley & Sons, New York, NY, (1988).

Liebowitz, Jay, ed., *Expert System Applications to Telecommunications*, John Wiley & Sons, New York, NY, (1988).

Liebowitz, Jay and Daniel De Salvo, ed., *Structuring Expert Systems*, Yourdon Press, Englewood Cliffs, NJ, (1989).

Oliff, Michael D., *Expert Systems and Intelligent Manufacturing*, Elsevier Science Publishing Co., Inc., New York, NY, (1988).

Shneiderman, Ben, *Designing the User Interface*, Addison-Wesley Publishing Co., Reading, MA, (1987).

Ullman, Jeffrey, *Principles of Database and Knowledge-Base Systems*, Computer Science Press, Rockville, MD, (1988).

Winston, Patrick H., and Karen A. Prendergast, ed., *The AI Business: Commercial Uses of Artificial Intelligence*, The MIT Press, Cambridge, MA, (1984).

Appendix A Overview of Multiattribute Utility

This appendix is adapted from the appendix in Volume 5, *TESTER_C User's Manual*, by Jacob W. Ulvila. It gives an overview of multiattribute utility analysis as applied to testing expert systems.

Multiattribute utility (MAU) analysis (Keeney and Raiffa, 1976) is a methodology that is grounded in the mathematics of measurement. MAU provides an appropriate procedure for evaluation in cases where multiple objectives are important. MAU models reflect explicitly the relative importance of various performance levels on different objectives and the tradeoffs among objectives.

The key stages in a multiattribute utility (MAU) analysis, as they relate to testing, are as follows:

- identification of what is to be evaluated (e.g., a particular expert system);
- definition of the criteria, factors, or attributes of value;
- evaluation, or "scoring" of systems against the attributes;
- prioritization of the attributes of value;
- evaluation of systems;
- sensitivity analyses.

Identification of What is to be Evaluated

The first step is to determine what is being evaluated. In the case of a system being tested, that system is to be evaluated. It is also important to identify more precisely whether the evaluation includes a particular hardware/software system, the system and its human operators, or something else. This choice will influence the choice of attributes and the scope of testing. For purposes of this illustration, we will assume that a single, well-defined hardware/software system is being tested and that these test results are being evaluated to determine the acceptability of the system to perform functions to assist human operators, but that the operators themselves are not being tested. We further assume for illustration that it is desirable for the test to identify areas of strength and weakness in the system as well as indicate its acceptability.

To use MAU analysis for test evaluation, it is also useful to construct additional hypothetical "benchmark systems" to use as points of reference. For purposes of this analysis, we will specify the following "systems":

- the test system, which is the system being subjected to testing;

- a goal system, which is a hypothetical system that fully attains every goal on every attribute;

- a failing system, which is a hypothetical system that fails on every attribute;

- a marginal system, which is a hypothetical system that, on balance, would just manage to pass the test, considering its performance over all attributes.

Introduction of these hypothetical systems enables a tester to apply the test criteria on a consistent, comparative basis, and to highlight areas of deficient and superlative performance. Of the hypothetical systems, the marginal one may be most difficult but most important to describe. Any given system under test is likely to have some areas where it falls short of goals and others where it exceeds goals. In addition, some of the goals may be set as ideals that could not be expected to be met. The marginal system provides a way for the tester to interpret performance in a way that recognizes these possibilities, and to specify *in advance* a minimal level of acceptable performance. This specification in advance removes some of the subjectiveness of the process by setting an overall level of acceptability before test results are known. Note that the marginal system will not generally be unique. Many possible combinations of performance against attributes may be minimally acceptable. However, when the MAU model is fully specified, all of these marginal systems should receive about the same overall evaluation. Specification of one of these systems thus aids in the overall evaluation of the actual system being tested.

Identifying Attributes of Value

Figure A.1 shows the attributes of value developed in *Volume 1: Handbook for Testing Expert Systems*. These attributes are defined at the end of this appendix.

Evaluation on Attributes

Next, a scale is developed for each bottom-level attribute that relates improvements on the scale to the value to the organization. Often this scale can be developed using natural standard units (e.g., minutes for time, percent correct for accuracy) when such units exist. The relationship between changes on the scale and the value of the changes is then established, and the value is transformed for modeling purposes into a standard scale, such as a 0 to 100 scale. In cases where no natural units exist, a relative value scale, such as a 0 to 100 point scale, could be used directly. Here, it is important to define the points on the scale carefully in terms of the attribute being represented so that unbiased assessments can be made.

For purposes of this report, we define the scales of utility such that a 0 represents failure on that attribute and 50 represents meeting fully the performance goal on the attribute. This choice is arbitrary in the sense that these levels of performance could be assigned any numbers, for example, 0 and 100, 0 and 1000, or 27 and 78. However, the points are not arbitrary in their meaning; 0 is assigned consistently to the failure level, and 50 is assigned consistently to the level of full satisfaction. This assignment provides a basis for consistent interpretation of the analysis and provides the kind of consistency that reduces bias from the assessments. The scale also allows value to be attached to performance that exceeds the goal, by scores greater than 50. Scales represent ratio judgments of value in the following manner. A score of 25 is halfway (in value) between failure and full goal

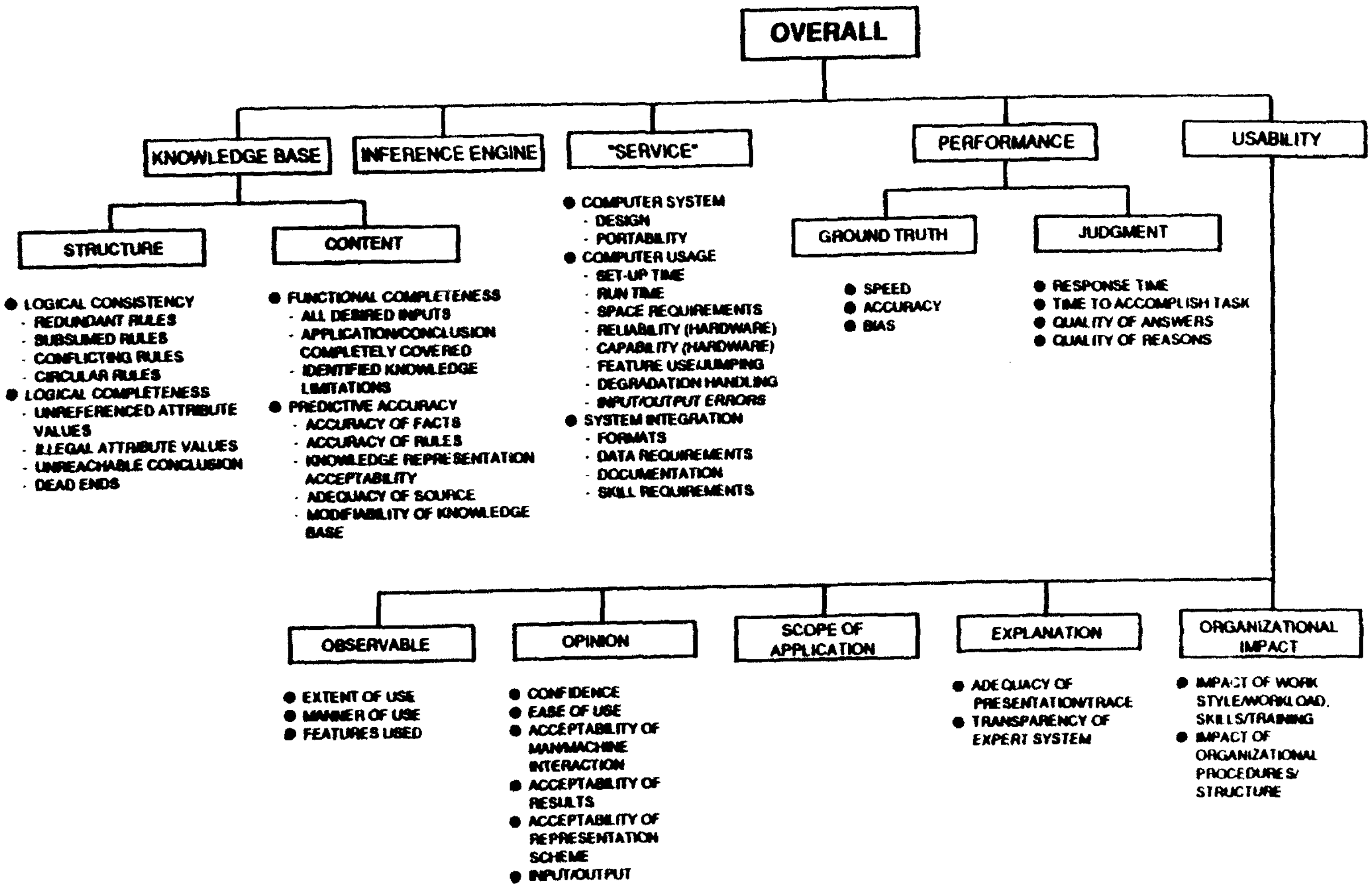

Figure A.1 A MAU Framework for Testing and Evaluating Expert Systems

attainment. This provides for convenient and consistent interpretation of scores. However, scores represent value on individual attributes only, and a score on one scale is not generally comparable to a score on another attribute. The weighting procedure described below provides a means for comparing across attributes.

Since a later step in the analysis makes comparisons across attributes, the method implies that attributes are, to some degree, compensatory. That is, a low score on some attribute can be compensated for, at least partially, by high scores on other attributes. If such is not the case, then a threshold should be established on that attribute. A threshold is a minimal level of performance on a single attribute that must be met. Failure to meet this performance renders the system under test unacceptable regardless of its performance in other areas.

Suppose the goal on set-up time were five minutes and that sixty minutes of set-up time were considered totally unacceptable. Suppose, further, that the shorter the set-up time, the better, with instantaneous set-up ideal and that an increase from 5 to 15 minutes was considered as serious as an increase from 15 to 60 minutes. This would result in the utility curve shown in Figure A.2.

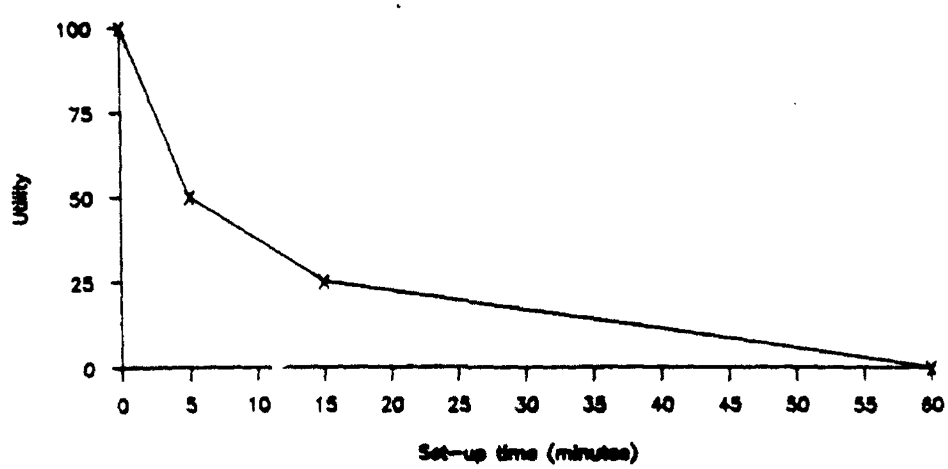

Figure A.2 Utility Curve for Set-Up Time

A utility curve reflects value and thus is inherently subjective. However, the explicitness with which these values are used in the MAU analysis removes the possibility of evaluation on the basis of a hidden agenda. The results and outcome of an MAU analysis are reproducible by people who share the same judgment over appropriate values and tradeoffs to use, and differences in evaluations by different people can be traced to specific differences in their value structures, which are open to inspection in the MAU analysis.

The utility curve could take on many different shapes. In some cases, utility increases slightly until some point is reached and then it rises dramatically. In other cases, utility is "all or nothing;" that is, no value is perceived until a certain point is reached, then all value is obtained. It is also possible for utility to rise up

to a target point and then drop off (e.g., for bias, which runs from -1 to + 1, with a target of 0). These situations could lead to the following types of curve:

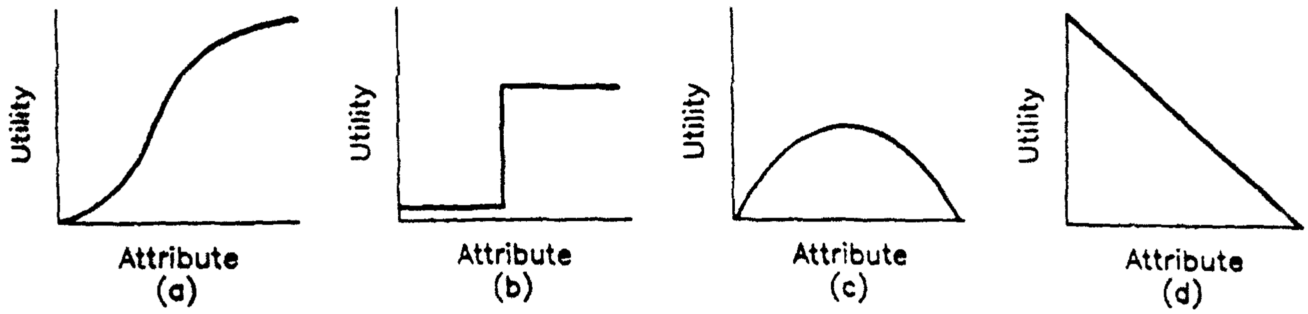

Figure A.3 Some Possible Shapes for Utility Curves

There is also no requirement that utility curves be continuous. Sometimes the attribute can be measured in discrete terms, or categories, even though there is a continuous range for the measure. An example is shown in Figure A.4.

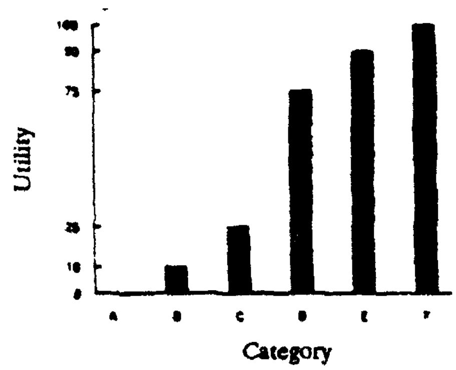

Figure A.4 Utility Curve for a Discrete Categorical Variable

Two important features of utility assignment are worth noting. First, the horizontal axis for each factor is determined uniquely for that factor. Common sense and logic dictate the appropriate measure. Second, it is not necessary to develop formally the utility curves themselves. Once the logic behind the curves is apparent, scores might be directly assessed.

Often, there is not a readily quantifiable measure for an attribute. In these cases, verbal descriptions of relative measure can be used. An example is the categorical attribute shown in Figure A.4. Scaling terms such as High/Medium/Low, Yes/No, Poor/Good/Very Good, and Go/No might also be used to define measurement scales.

Prioritization of the Attributes (Weighting)

In the scoring systems described above, an evaluation scale from 0 to 100 was developed for each factor. However, each scale is defined independently of all others, and the resulting scores are not directly comparable. In any real test, some attributes carry more importance than others, and a measure of the priority, or relative importance, of each factor is necessary for an overall evaluation. This is accomplished through a weighting system. As with the scores, weights are judgments, and could vary from organization to organization or from tester to tester. MAU analysis makes such weights explicit, however, and available for review. (Weights should also be subjected to sensitivity analyses as discussed below.)

The most common perception of a weight is that it answers the question, "How *important* is attribute A relative to attribute B?" Unfortunately, such a question often obscures the issue of evaluation. A more pertinent question to ask is, "How important is the *difference* along the range in values for attribute A versus the *difference* for attribute B?" The subtle differences in wording of these two questions is extremely important. The latter question includes both the importance of the attribute as well as the "swing" in the range of values on the attributes. The interpretation of weights and the procedure for assessing them depends on the form of the model to be employed to aggregate the single-attribute scales. The theoretical basis for a variety of aggregation models has been developed (see Keeney and Raiffa, 1976), but an additive aggregation rule is appropriate or a sufficient approximation in most cases. (The additive rule is appropriate if "additive independence" conditions are met.)

Weighting can be accomplished top-down or bottom-up. Top-down weighting is usually easier. In the top-down approach, the analyst begins at the highest-level node in the hierarchy and assesses the relative differences among attributes. A common approach assigns a weight of 100 to the most important swing. Other weights are then assigned using ratio judgments—that is, if the swing on an attribute is judged to be twice as important as the swing on another attribute, the former would carry twice the weight of the latter. Furthermore, with additive independence, weights can be compared in an additive sense. If attribute A is weighted at 100, attribute B at 75, and attribute C at 50, this implies that, for example, the combined effect of 50-point swings in both B and C (added weights equal 125) is more important than a 50-point swing on A (weight of 100). Such comparisons can serve as a good calibration check on the weights.

For consistency in the analysis, the weights are often normalized to sum to 1.00 by adding the assigned weights and dividing each by the sum. The basis for assigning weights might be in a statement of requirements or other guidance provided to the tester, and this could well vary from one test to another. See Chapter 7 of *Volume 1: Handbook for Testing Expert Systems* for more information. In fact, some guidance may be such that some of the attributes are irrelevant. If this is the case, a weight of zero could be assigned to the attribute. In any case, assigning weight *before* the test is conducted is a recommended procedure to reduce bias.

Evaluation

After scores have been assessed against the attributes and weights have been assigned, evaluations can be determined. Since an additive MAU analysis is used, the overall evaluation of an alternative is a weighted average of assessed scores, with the exception that a system is regarded as a failure if any score falls below a threshold on any attribute. (This is described in more detail in *Volume 1: Handbook on Testing Expert Systems.*)

Although the numerical results of a MAU analysis provide a compact representation of the evaluation, they are not the only results of the analysis. The numerical output can also direct the tester to areas of strength and weakness in the system under test and thus provide the basis for suggested improvements. The numerical output also summarizes explicitly the judgments used and thus provides a basis for building a verbal

case for or against the system. Also, the explicit numerical representations of judgment, especially in the form of weights, provide a means of identifying important differences of opinion if differences exist between testers and evaluators.

Sensitivity Analysis

Several reasons recommend sensitivity analyses for most test evaluations. First, some parts of the analysis may not be known with a high degree of accuracy for any of a number of reasons. While it is desirable to design and conduct tests that provide highly accurate assessments, lack of resources or other reasons sometimes prevent the level of accuracy desired. At this point, the test evaluator may decide to include or exclude the data from consideration in the evaluation. We recommend including the data but running sensitivity analyses over the range of uncertainty. Use of judgmental information is another reason to perform sensitivity analyses.

There are three major types of sensitivity analyses. First, the scores that have been assessed can be modified to determine if results change. This type of sensitivity analysis is appropriate in cases where scores were assessed with inadequate test data or where judgmental assessments were made. Generally, results are reasonably insensitive to minor changes in scores, especially with an analysis that uses the whole MAU framework. Next, several weights can be changed and the overall scores recalculated. This is useful in examining large-scale changes to the model (such as using weights for a different evaluator), but does not make it easy to isolate causes of change or disagreement. A third sensitivity analysis is to vary one weight at a time and identify the regions where evaluations change. Typically, one factor is chosen and its weight is allowed to vary over a wide range. As the weight increases, the total weight of the other factors must decease but the weights are kept in the same relative proportion to each other.

Attribute Definitions

Figure A.1 shows our MAU proposed framework for testing and evaluating expert systems. The overall assessment of the expert system is composed of five criteria: knowledge base, inference engine, service requirements, performance, and usability. These are subdivided to the level of attributes as described below.

KNOWLEDGE BASE. These attributes refer to the structure and content of the expert system's knowledge base. While the descriptions below are phrased in terms of a rule base, analogous attributes would apply to a frame-based expert system. (See Hayes, 1981, for a discussion of the logical equivalents of rule-based and frame-based systems.)

<u>Structure</u>

Logical Consistency. The following attributes would limit the consistency (or correspondence) and efficiency of a knowledge base. Redundant rules are rules or groups of rules that have essentially the same conditions and conclusions. Redundancy can be due to duplicate rules or the creation of equivalent rules (rule groups) by wording variations in the names given to variables, or the order in which they are processed. Subsumed rules occur when one rule's (or group of rules') meaning is already expressed in another rule (or group of rules) that reaches the same conclusion from similar but less restrictive conditions. Conflicting rules are rules (or groups of rules) that use the same conditions, but result in different conclusions, or rules whose combination violates principles of logic (e.g., transitivity). Circular rules are rules that lead one back to an initial (or intermediate) condition instead of a conclusion.

Logical Completeness. A knowledge base is complete if it has no holes or gaps in its logic. The following attributes indicate a logical incompleteness. Unreferenced attribute values are values on a condition that are not defined; consequently, their occurrence cannot result in a conclusion. Illegal attribute values are values

on a condition that are outside the acceptable set or range of values for that condition. An unreachable conclusion is a conclusion that cannot be triggered by the rules combining conditions. Dead ends are rules that do not connect input conditions with output conclusions.

Content

Functional Completeness is the extent to which the knowledge base addresses all domain conditions. All desired inputs: the knowledge base can handle all input conditions that need to be addressed. Application/conclusion completely covered: the knowledge base can trigger all output conclusions that need to be addressed. Identified knowledge limitations: the rules in the knowledge base can tell the user if input conditions currently being processed cannot be addressed. Analogously, if the expert system is such that a user can specify a conclusion in order to identify the input conditions that would generate it (e.g., as in a backward-chaining system), an expert system that was knowledgeable of its limitations would tell users if a conclusion currently being processed as input could not be addressed.

Predictive Accuracy. The following attributes address the accuracy and adequacy of the knowledge base. Problems here may also be related to problems of performance. Accuracy of facts: the quality of the unconditional statements in the knowledge base. Accuracy of rules: the quality of the conditional statements in the knowledge base representing expert judgment. Knowledge representation acceptability: whether or not the scheme for representing knowledge is acceptable to other domain experts and knowledge engineers. Adequacy of source: the quality of the persons or documentation used to create the knowledge base. Modifiability of knowledge base: the extent to which the knowledge base can be changed and the control over that change.

INFERENCE ENGINE: the extent to which the inference engine provides error-free propagation of rules, frames, probabilities, or other representation of knowledge or uncertainties used in the system.

"SERVICE" refers to aspects of the system (computer and others) in which the expert will operate.

Computer System. Design: the extent to which the expert system runs on the approved computer hardware and operating system and utilizes the preferred complement of equipment and features. In some cases, the design system will be stated in a requirements document; in other cases, the tester may need to survey available equipment at the intended installation. Portability: how easily the expert system can be transferred to other computer systems.

Computer Usage. Set-up time: the amount of time required for the computer operator to locate and load the program (if any) and the time to activate the program. Set-up time should be measured under the expected operating conditions. Run time: the amount of time required to run the program with a realistic set of input data. This attribute refers only to the time that the computer program takes to run; the time needed for the user is under PERFORMANCE factors. Space requirements: the amount of RAM, disk, or other space required by the program. Hardware reliability: the percentage of time the computer system could be expected to be operating effectively. Hardware capability: the computer system's total amount of RAM, disk, or other space. Effect of feature use/jumping: the extent to which moving from various parts of the program causes errors. Degradation: how well the program saves data and analyses and permits continuation after an unexpected program or system crash or power outage. Handling input errors: the extent to which the program prohibits a program crash and tells the user what to do after an input mistake.

System Integration. Formats: the extent to which the program uses input and output formats that are consistent with the intended use. This includes any mandated or standard formats that are specific to the intended user organization. Data requirements: the extent to which the program's data requirements are consistent in content, quantity, quality, and timeliness with those available to the intended user organization. The expert system should also be able to interact with specified and appropriate databases and communications

systems. Documentation: the adequacy of material regarding the program's use and maintenance. Copies of computer code and its supporting documentation should be complete and understandable, and should allow maintenance by the user organization. (All applicable software documentation standards should be met.) Skill requirements: - the extent to which the program can be operated by appropriately skilled individuals. The appropriate skill requirement includes grade level (for military enlisted, military officer, or civilian personnel), users' technical background, and training requirements. The appropriate level may be specified in requirements or may be determined by reference to the organizational setting of its intended use and to the personnel assigned to that setting.

PERFORMANCE refers to the operation of the expert system and the user. It includes both comparisons with ground truth and judgmental assessments.

Performance against Ground Truth. Speed: the amount of time it takes a user working with the expert system to solve representative problems. Accuracy: the degree of overlap in the distributions of belief values when the hypothesis is true versus false (see Chapter 5 of Volume 1, *Handbook for Testing Expert Systems*). Bias: the difference in the proportion of false negatives (hypothesis is true but system says false) to false positives (hypothesis is false, but system say it's true) (see Chapter 5 of Volume 1, *Handbook for Testing Expert Systems*).

Judgmental Performance. Response time: the judgments of users regarding the adequacy of the amount of time the expert system takes to react to inputs. Time to accomplish task: the judgments of users regarding the adequacy of the amount of time required to perform the task when using the expert system. Quality of answers: the judgments of users and experts regarding the system's capability. Quality of reasons: the judgments of users and experts regarding the adequacy of the system's justification for its answers.

USABILITY is the extent to which the expert system, or parts of the expert system, is used, is acceptable to individuals, and is acceptable to the organization.

Observable Usability includes aspects of usability that a tester can observe (or a system can record) during a test without asking the test subjects. Extent of use: how much users employ the expert system to perform the task (e.g., the proportion of time that the system was used to accomplish tasks assigned in a test). Manner of use: the way in which users employ the system and its features, including the procedures to access different modules, the way that intermediate and final outputs are incorporated into the user's results, and the use of interfaces. Features used: the extent to which different aspects of the expert system are employed by users.

Opinions about Usability. Confidence: how confident users feel in taking actions based on working with the expert system. Ease of use: how easy users judge the system is to use after they have completed training and become familiar with the system. Acceptability of person/machine interaction process: the extent to which users assess that they and the system are performing the tasks or activities for which they are best suited. Acceptability of results: the users' judgments regarding the adequacy of the system's capability. Acceptability of representation scheme: the users' judgments regarding the adequacy of the system's way of presenting knowledge. Input/output: the user's judgment about the adequacy of the extent, display, and manner of accessing the expert system's input and output.

Scope of Application: the users' judgments regarding the adequacy of the expert system in addressing domain problems.

Explanation. Adequacy of presentation and trace: the users' judgments regarding the acceptability of the system's presentation of its reasoning process. Transparency of expert system: the extent to which the system's reasoning process is clear and understandable to its users.

Organizational Impact. Impact on work style, workload, skills, and training: the judgments of users regarding the impact of the expert system on how they do their job, or the skills and training required to perform it effectively. Impact on organizational procedures and structure: the judgments of users regarding the impact of the expert system on the organization's operations.

References

Hayes, P.J. (1981). "The Logic of Frames." B.L. Webber and N.J. Nilsson (Eds.), *Readings in Artificial Intelligence*. Palo Alto, CA: Tioga, 451-458.

Keeney, R.L. and H. Raiffa (1976). *Decisions with Multiple Objectives*. New York: Wiley.

CONCEPT PHASE CHECKLIST

APPLICABILITY OF PROBLEM AREA

			YES (1pt.)	NO (0pt.)	
C.1	SUBJECT AREA WELL DEFINED	≥5	[ ]	[ ]	<5
C.2	DECISION-BASED PROCESS	≥3	[ ]	[ ]	<3
C.3	EXPERT SYSTEM TECHNOLOGY MOST APPROPRIATE	≥4	[ ]	[ ]	<4
C.4	EXPERT AVAILABLE	≥4	[ ]	[ ]	<4
C.5	NEED FOR KNOWLEDGE DISTRIBUTION	≥4	[ ]	[ ]	<4

Total: ______

BENEFITS TO BE ACHIEVED

	LOW 1	2	3	4	HIGH 5
IMPROVED PRODUCTIVITY	___	___	___	___	___
IMPROVED EFFICIENCY/TIME SAVINGS	___	___	___	___	___
IMPROVED ACCURACY	___	___	___	___	___
IMPROVED CONSISTENCY	___	___	___	___	___
IMPROVED TRAINING	___	___	___	___	___
IMPROVED INFORMATION HANDLING	___	___	___	___	___
REDUCED DOWNTIME	___	___	___	___	___
OTHER ISSUES ______________	___	___	___	___	___

Total: ______

COSTS TO BE INCURRED

	LOW 5	4	3	2	HIGH 1
TIME AND LABOR	___	___	___	___	___
HARDWARE AND SOFTWARE	___	___	___	___	___
SYSTEM TESTING	___	___	___	___	___
SYSTEM DISTRIBUTION AND LICENSING	___	___	___	___	___
SYSTEM MAINTENANCE	___	___	___	___	___
UPGRADE MANAGEMENT AND VERSION CONTROL	___	___	___	___	___
OTHER ISSUES ______________	___	___	___	___	___

Total: ______

SUPPORT FOR EXPERT SYSTEM

	LOW 1	2	3	4	HIGH 5
DEVELOPERS	___	___	___	___	___
END-USERS	___	___	___	___	___
MANAGEMENT	___	___	___	___	___

Total: ______

MILESTONE

		YES	NO	
WILL EXPERT SYSTEM TECHNOLOGY BE USED?	≥55	[ ]	[ ]	<55

CONCEPT PHASE WORKSHEET

C.1 WELL-DEFINED SUBJECT AREA

	YES (1PT.)	NO (0PT.)
DATA IS STRUCTURED	☐	☐
BOUNDARIES ARE CLEAR	☐	☐
PEOPLE HAVE KNOWLEDGE IN AREA	☐	☐
KNOWLEDGE IS AT RIGHT STAGE OF MATURITY	☐	☐
SUBJECT IS NARROW & ISOLATED	☐	☐
SUBJECT CAN BE DIVIDED INTO STAND-ALONE SEGMENTS	☐	☐
INCREMENTAL PROGRESS IS POSSIBLE	☐	☐

TOTAL:______

C.2 DECISION-BASED PROCESS

	YES (1PT.)	NO (0PT.)
REQUIRES QUALITATIVE OR SUBJECTIVE REASONING	☐	☐
TASK IS COGNITIVE	☐	☐
TASK HAS MANY POSSIBILE COMBINATIONS	☐	☐
TASK INVOLVES CHAINS OF REASONING	☐	☐

TOTAL:______

C.3 EXPERT SYSTEM TECHNOLOGY MOST APPROPRIATE

	YES (0PT.)	NO (1PT.)
QUANTITATIVE REASONING	☐	☐
DATA STORAGE & RETRIEVAL	☐	☐
WORD PROCESSING	☐	☐
MODELS & SIMULATIONS	☐	☐
PROCEDURAL PROGRAMMING	☐	☐
RAPIDLY CHANGING DATA	☐	☐

TOTAL:______

CONCEPT PHASE WORKSHEET (cont.)

C.4 EXPERT AVAILABLE

	YES (1PT.)	NO (0PT.)
EXPERT EXISTS (BETTER THAN AMATEURS)	☐	☐
EXPERT CAN BE IDENTIFIED	☐	☐
EXPERT IS WILLING OR EAGER	☐	☐
EXPERT IS AVAILABLE AND ACCESSIBLE	☐	☐
EXPERT CAN COMMUNICATE DETAILED KNOWLEDGE	☐	☐

EXPERT'S NAME________________________________ TOTAL:______

C.5 NEED FOR KNOWLEDGE DISTRIBUTION

	YES (1PT.)	NO (0PT.)
OTHERS COULD BENEFIT FROM KNOWLEDGE	☐	☐
PROCESS WOULD BE IMPROVED	☐	☐
NEED FOR KNOWLEDGE TRANSFER	☐	☐
PROBLEM WITH EXPERT ACCESSIBILITY	☐	☐
NEED FOR TRAINING	☐	☐
GEOGRAPHIC DISTRIBUTION PROBLEM	☐	☐
IMPROVEMENTS NEEDED IN DECISION-MAKING PROCESS	☐	☐

TOTAL:______

DEFINITION PHASE CHECKLIST

SCOPE AND REQUIREMENTS DEFINED*

DATE

D.1 REQUIREMENTS STATEMENT COMPLETED ☐ ____________

DESIGN OF EXPERT SYSTEM PREPARED*

D.2 ONE-PAGE DESIGN COMPLETED ☐ ____________

D.3 DETAILED DESIGN COMPLETED ☐ ____________

DEVELOPMENT TEAM SELECTION*

EXPERT ______________________________

DEVELOPER(S) ______________________________

MANAGER ______________________________

OUTSIDE REVIEWER ______________________________
(IF APPLICABLE)

KNOWLEDGE ACQUISITION APPROACH SELECTION

(INDICATE DEGREE OF USE)

ELICITING KNOWLEDGE — LOW → HIGH

STRUCTURED INTERVIEWS ___ ___ ___ ___ ___

UNSTRUCTURED INTERVIEWS ___ ___ ___ ___ ___

QUESTIONNAIRES ___ ___ ___ ___ ___

OBSERVATION ___ ___ ___ ___ ___

DOCUMENTS ___ ___ ___ ___ ___

DOCUMENTING KNOWLEDGE

DECISION TREES ___ ___ ___ ___ ___

TABLES ___ ___ ___ ___ ___

KNOWLEDGE MAPS ___ ___ ___ ___ ___

END USER CONTACT

LOW → HIGH

LEVEL OF INTEREST ___ ___ ___ ___ ___

UNDERSTANDING OF SUBJECT AREA ___ ___ ___ ___ ___

LEVEL OF COMPUTER EXPERTISE ___ ___ ___ ___ ___

COMMENTS ____

____

* MILESTONES

D.1 REQUIREMENTS STATEMENT

SUBJECT AREA ______________________________

SCOPE (SPECIFIC BOUNDARIES WITHIN SUBJECT AREA)

____

____

SYSTEM REQUIREMENTS (HARDWARE, SOFTWARE, ETC.)

____

____

PURPOSE (ROLE IN PROCESS, EXPECTED RESULTS, ETC.)

____

____

____

INTENDED USERS (WHO WILL USE IT)

____

____

CAPABILITIES (WHAT IT WILL DO)

____

____

LIMITATIONS (WHAT IT WILL NOT DO)

____

____

OUTPUTS (FILE INTERACTIONS, DEPENDENCIES, REPORTS, ETC.)

____

____

EXPECTATIONS (BENEFITS, IMPROVEMENTS, ETC.)

____

____

ROUTINE MAINTENANCE AREAS

____

____

D.2 ONE-PAGE DESIGN

SYSTEM OVERVIEW: ______________________________

____

____

SYSTEM COMPONENTS

____

____

____

BASIC SYSTEM STRUCTURE (ATTACH APPLICABLE DRAWINGS)

____

____

____

SYSTEM INTERFACES

____

____

____

EXTERNAL CALLS AND DATA SOURCES

____

____

____

KNOWLEDGE REPRESENTATION

____

____

INFERENCING MECHANISM

____

____

D.3 DETAILED DESIGN

COMPLETE THIS FORM FOR EACH SYSTEM COMPONENT

COMPONENT: ____________________

PURPOSE: ____________________

SCOPE: ____________________

NECESSARY INFORMATION: ____________________

SPECIFIC EXTERNAL CALLS: ____________________

SPECIFIC DATA SOURCES: ____________________

PROTOTYPE DEVELOPMENT PHASE CHECKLIST

DEVELOPMENT TASKS

			DATE(S)
P.1	EXPERT SYSTEM SHELL SELECTED	☐	________
P.2	USERS CONSULTED ON INTERFACE ISSUES	☐	________
P.3	DOMAIN ORIENTATION COMPLETED	☐	________
P.4	KNOWLEDGE ACQUISITION SESSIONS COMPLETED	☐	________
P.5	CASES SET ASIDE FOR TESTING	☐	________
	KNOWLEDGE CONVERTED INTO KNOWLEDGE REPRESENTATION	☐	________
	INCREMENTAL TESTING PERFORMED	☐	________
P.6	REQUIREMENTS STATEMENT REVIEWED	☐	________
P.7	DESIGN DOCUMENTS REVIEWED	☐	________
P.8	DOCUMENTATION PREPARED	☐	________

MILESTONE

PROTOTYPE EXPERT SYSTEM PRODUCED	☐	________

PROTOTYPE PHASE WORKSHEET

P.1 EXPERT SYSTEM SHELL SELECTION

SELECTION CRITERIA:	RANK
AVAILABILITY TO ORGANIZATION	____
KNOWLEDGE REPRESENTATION	____
INFERENCE ENGINE	____
USER AND DEVELOPER INTERFACE	____
INTEGRATION WITH OTHER PROGRAMS/FILES	____
COST	____
EASE OF DISTRIBUTION	____
HARDWARE REQUIREMENTS	____
MAINTENANCE ISSUES	____
OTHERS: ______________________________	____
______________________________	____

P.1.1 EVALUATION MATRIX COMPLETED ☐

SHELL SELECTED: ______________________

P.2 USERS CONSULTED ON INTERFACE ISSUES

DESIRABLE FEATURES:	RANK
MENUS	____
GRAPHICS	____
FUNCTION KEYS	____
REPORTS	____
FILE MANAGEMENT	____
HELP FACILITIES	____
SIMILARITY TO CURRENT SYSTEMS	____
JUSTIFICATION CAPABILITIES	____
OTHERS: ______________________________	____
______________________________	____
______________________________	____

P.1.1 EXPERT SYSTEM SHELL EVALUATION MATRIX

CRITERIA / SHELL					

PROTOTYPE PHASE WORKSHEET (cont.)

P.3 DOMAIN ORIENTATION

SOURCE OF BACKGROUND DATA IDENTIFIED ☐

SOURCES USED: ______________________________

____

____

EXPERT(S) OBSERVED AT WORK ☐

GENERAL CONVERSATIONS WITH EXPERT(S) ☐

P.3.1 FINDINGS RECORDED ☐

P.4 KNOWLEDGE ACQUISITION SESSIONS

SCHEDULE DEVELOPED ☐

EXPERT BRIEFED ON PROJECT ☐

KNOWLEDGE ACQUISITION PLANNER COMPLETED FOR EACH SESSION ☐

P.5 TEST CASE SELECTION

TEST CASES IDENTIFIED

AVERAGE/FREQUENT CASES ☐

EXTREME/SIGNIFICANT CASES ☐

TEST CASES SET ASIDE FOR LATER TESTING ☐

P.3.1 DOMAIN ORIENTATION

DATE: ______________________

REFERENCE SOURCE: ______________________

IMPORTANT DATA: ______________________

DATE: ______________________

REFERENCE SOURCE: ______________________

IMPORTANT DATA: ______________________

P.3.1 DOMAIN ORIENTATION (cont.)

DATE: ____________

EXPERT: ____________

OBSERVATIONS: ____________

COMMENTS: ____________

KNOWLEDGE ACQUISITION PLANNER
(COMPLETE FOR EACH SESSION)

DATE: ____________ **INTERVIEW NO.:** ____________

- INTERVIEW GUIDE DEVELOPED ☐
- INTERVIEW PERFORMED ☐
- KNOWLEDGE RECORDED ☐
- KNOWLEDGE COMPILED ☐
- MATRICES, DECISION TREES DEVELOPED ☐
- KNOWLEDGE ACQUIRED TO DATE ANALYZED ☐
- PROBLEMS IN KNOWLEDGE IDENTIFIED ☐

PROBLEMS IDENTIFIED: ____________

ISSUES FOR NEXT INTERVIEW: ____________

DATE AND TIME OF NEXT INTERVIEW: ____________

INTERVIEW GUIDE

DATE: ____________________ INTERVIEW NO.: ____________________

EXPERT: ____

KNOWLEDGE ENGINEER: ______________________________

TOPICS TO BE COVERED: ______________________________

ISSUES TO BE RESOLVED: ______________________________

NEW KNOWLEDGE: ____

PROTOTYPE PHASE WORKSHEET (cont.)

P.6 REQUIREMENTS STATEMENT REVIEWED

- REQUIREMENTS REMAIN FEASIBLE ☐
- PROJECT STILL CONFORMS TO REQUIREMENTS ☐
- CONFLICTS IDENTIFIED ☐

____

____

____

- DOCUMENT MODIFIED ACCORDINGLY ☐

P.7 DESIGN DOCUMENTS REVIEWED

- DESIGN REMAINS ACCURATE ☐
- PROJECT STILL CONFORMS TO DESIGN ☐
- CONFLICTS IDENTIFIED ☐

____

____

____

- DOCUMENTS MODIFIED ACCORDINGLY ☐

P.8 DOCUMENTATION PREPARED

USER'S MANUAL

- BACKGROUND/OVERVIEW ☐
- INTENDED USERS ☐
- ACCESSING SYSTEM ☐
- INTERFACE FORMAT ☐
- USING THE SYSTEM ☐
- FILES INVOLVED ☐
- SAMPLE SESSIONS ☐

DEVELOPER'S NOTEBOOK

- REQUIREMENTS STATEMENT ☐
- SYSTEM DESIGN ☐
- DEVELOPMENT WORKSHEETS ☐
- KNOWLEDGE ACQUISITION DATA ☐
- MATRICES, GRAPHICS ☐
- FILE PROCESSING ☐
- MAINTENANCE ISSUES ☐
- PROJECT MANAGEMENT AIDS ☐

TEST AND EVALUATION PHASE CHECKLIST

VERIFICATION BY EXPERT

PERSON RESPONSIBLE ______________________

		DATE
TESTING COMPLETED	☐	______________
FINDINGS RECORDED	☐	______________
E.1 EXPERT EVALUATION FORM COMPLETED	☐	______________

VERIFICATION BY KNOWLEDGE ENGINEER

PERSON RESPONSIBLE ______________________

TESTING COMPLETED	☐	______________
FINDINGS RECORDED	☐	______________
E.2 DEVELOPER EVALUATION FORM COMPLETED	☐	______________

VALIDATION OF OVERALL SYSTEM BY USERS

TEST DISTRIBUTED	☐	______________
TESTING COMPLETED	☐	______________
FINDINGS RECORDED	☐	______________
E.3 USER'S EVALUATION FORMS COMPLETED	☐	______________

EVALUATION OF USER'S MANUAL

E.4 USER'S MANUAL EVALUATION FORMS COMPLETED	☐	______________

EVALUATION OF DEVELOPER'S NOTEBOOK

E.5 DEVELOPER'S NOTEBOOK EVALUATION FORMS COMPLETED	☐	______________

REQUIREMENTS AND DESIGN REVIEWED

E.6 REQUIREMENTS EVALUATION COMPLETED	☐	______________
E.7 DESIGN EVALUATION COMPLETED	☐	______________

MILESTONE

TESTING AND EVALUATION COMPLETED	☐	______________

E.1 EXPERT EVALUATION FORM

EXPERT: ______________________ **DATE:** ______________

TEST CASES APPLIED TO EXPERT SYSTEM

		NUMBER
AVERAGE/FREQUENT CASES	☐	________
EXTREME/SIGNIFICANT CASES	☐	________

PROBLEMS IDENTIFIED

KNOWLEDGE ____

____

____

____

LOGIC ____

____

____

____

ORDER OF QUESTIONING ___________________________________

____

____

____

OVERALL IMPRESSION LOW → HIGH

___ ___ ___ ___ ___

COMMENTS ____

____

____

____

____

E.2 DEVELOPER EVALUATION FORM

NAME: ______________________ **DATE:** ______________

TEST CASES APPLIED TO EXPERT SYSTEM

		NUMBER
AVERAGE/FREQUENT CASES	☐	________
EXTREME/SIGNIFICANT CASES	☐	________

PROBLEMS IDENTIFIED

LOGIC ____

INFERENCING ____

INTERFACE ____

COMMENTS ____

E.3 USER EVALUATION FORM

NAME: ______________________ **DATE:** ______________

COMMENTS

INTERFACE ____

____

____

____

RESPONSE TIME ____

USER-FRIENDLINESS ____

____

____

____

APPROPRIATENESS OF SUBJECT LEVEL ____

____

FILE MANAGEMENT ISSUES ____

____

GENERAL ____

____

____

____

OVERALL IMPRESSION LOW → HIGH

___ ___ ___ ___ ___

E.4 USER'S MANUAL EVALUATION FORM

NAME: ______________________ **DATE:** ______________

OVERALL DOCUMENT

	YES	NO
READABLE	☐	☐
ACCURATE	☐	☐
CLEAR	☐	☐
COMPLETE	☐	☐
EASILY UNDERSTANDABLE	☐	☐

COMMENTS ____

____

____

____

SAMPLE SESSIONS

	YES	NO
UNDERSTANDABLE	☐	☐
ACCURATE	☐	☐
CLEAR	☐	☐
COMPLETE	☐	☐
DESCRIPTIVE	☐	☐

COMMENTS ____

____

____

____

OVERALL IMPRESSION

LOW → HIGH

___ ___ ___ ___ ___

E.5 DEVELOPER'S NOTEBOOK EVALUATION FORM

NAME: ______________________ **DATE:** ______________

OVERALL DOCUMENT

	YES	NO
READABLE	☐	☐
ACCURATE	☐	☐
CLEAR	☐	☐
COMPLETE	☐	☐
EASILY UNDERSTANDABLE	☐	☐

COMMENTS ____

____

____

____

SUPPORTING DOCUMENTS INCLUDED

	YES	NO
REQUIREMENTS STATEMENT	☐	☐
SYSTEM DESIGN DOCUMENTS	☐	☐
DEVELOPMENT WORKSHEETS	☐	☐
KNOWLEDGE ACQUISITION MATERIALS	☐	☐
FILE PROCESSING AND MAINTENANCE ISSUES	☐	☐
PROJECT MANGEMENT AIDS	☐	☐

COMMENTS ____

____

____

____

OVERALL IMPRESSION LOW → HIGH

___ ___ ___ ___ ___

FINAL DEVELOPMENT PHASE CHECKLIST

		DATE COMPLETED
EVALUATION RESULTS COLLECTED		
END-USERS	☐	________
EXPERT	☐	________
DEVELOPER(S)	☐	________
OTHERS	☐	________
END-USER COMMENT EVALUATION		
COMMENTS COMPILED	☐	________
COMMENTS ANALYZED	☐	________
RELEVANT & FEASIBLE COMMENTS SELECTED	☐	________
CHANGES INCORPORATED INTO EXPERT SYSTEM		
END-USERS	☐	________
EXPERT	☐	________
DEVELOPER(S)	☐	________
OTHERS	☐	________
EXPERT SYSTEM RETESTED		
RULES VERIFIED	☐	________
LOGIC VALIDATED	☐	________
INTERFACE REVIEWED	☐	________
CHANGES INCORPORATED INTO DOCUMENTATION		
END-USERS	☐	________
KNOWLEDGE ENGINEER	☐	________
OTHERS	☐	________
MILESTONE		
FINAL EXPERT SYSTEM PRODUCED	☐	________

POST-DEVELOPMENT PHASE CHECKLIST

MAINTENANCE/TECHNICAL SUPPORT/UPGRADES

PERSON RESPONSIBLE ______________________

	YES	NO
HOTLINE REQUIRED	☐	☐

FREQUENCY OF REVIEW ______________________

SYSTEM UPGRADES

REASON	DATE
______________________	______________________
______________________	______________________
______________________	______________________
______________________	______________________

TRAINING (SELECT ALL THAT APPLY)

PERSON RESPONSIBLE ______________________

INDIVIDUAL	☐	TUTORIAL DISK	☐
GROUP	☐	DEMONSTRATIONS	☐
TRAINING MANUA	☐		

DISTRIBUTION (SELECT ALL THAT APPLY)

PERSON RESPONSIBLE ______________________

USER DATABASE PREPARED	☐	RUN-ONLY PREPARED	☐
USERS CONTACTED	☐	SYSTEM REPRODUCED	☐
DISKS ORDERED	☐	COVER LETTER PREPARED	☐
NUMBER OF 3 1/2"	____	DOCUMENTATION PRODUCED	☐
NUMBER OF 5 1/4" DD	____	PACKAGING PREPARED	☐
NUMBER OF 5 1/4" HD	____	REGISTRATION NOs ASSIGNED	☐
DISKS FORMATTED	☐		
LABELS PRODUCED	☐		

MILESTONE

FINAL SYSTEM IMPLEMENTED ☐ DATE: ______________________